SLAC/AP–103
April 1997

LIAR - A Computer Program for the Modeling and Simulation of High Performance Linacs*

R. Assmann, C. Adolphsen, K. Bane, P. Emma,
T. Raubenheimer, R. Siemann, K. Thompson, F. Zimmermann

Stanford Linear Accelerator Center, Stanford University,
Stanford, California 94309 U.S.A.

Version 1.9 – April 1997

WWW-homepage at
http://www.slac.stanford.edu/grp/arb/rwa/liar.htm

*Work supported by Department of Energy contract DE–AC03–76SF00515.

ABSTRACT

The computer program LIAR ("LInear Accelerator Research code") is a numerical modeling and simulation tool for high performance linacs. Amongst others, it addresses the needs of state-of-the-art linear colliders where low emittance, high-intensity beams must be accelerated to energies in the 0.05-1 TeV range. LIAR is designed to be used for a variety of different projects. It has been applied to and checked against the existing Stanford Linear Collider (SLC), the linacs of the proposed Next Linear Collider (NLC) and the proposed Linac Coherent Light Source (LCLS) at SLAC.

LIAR allows the study of single- and multi-particle beam dynamics in linear accelerators. It calculates emittance dilutions due to wakefield deflections, linear and non-linear dispersion and chromatic effects in the presence of multiple accelerator imperfections. Both single-bunch and multi-bunch beams can be simulated. It is possible to simultaneously study the acceleration of positive and negative charges in a linac. Diagnostic and correction devices include beam position monitors, RF pickups, dipole correctors, magnet movers, beam-based feedbacks, multi-device knobs and emittance bumps. Several basic and advanced optimization schemes are implemented. Present limitations arise from the incomplete treatment of bending magnets and sextupoles.

A major objective of the LIAR project is to provide an open programming platform for the accelerator physics community. We invite interested scientists to join this project. Due to its design, LIAR allows straight-forward access to its internal FORTRAN data structures. The program can easily be extended and its interactive command language ensures maximum ease of use. Presently, versions of LIAR are compiled for UNIX and MS Windows operating systems. An interface for the graphical visualization of results is provided. Scientific graphs can be saved in the PS and EPS file formats. In addition a Mathematica interface has been developed. LIAR now contains more than 40,000 lines of source code in more than 130 subroutines.

This report describes the theoretical basis of the program, provides a reference for existing features and explains how to add further commands. The LIAR home page and the ONLINE version of this manual can be accessed under:

http://www.slac.stanford.edu/grp/arb/rwa/liar.htm

1 Contents

2 THE LIAR PROJECT

The LIAR ("LInear Accelerator Research code") project was started at SLAC in August 1995 in order to provide a computing and simulation tool that addresses the needs of high performance linear accelerators. Its first objective was to implement advanced simulations for the main linacs of SLC (50 GeV) and NLC (500 GeV) at SLAC. Since then it has been applied to the LCLS project at SLAC (15 GeV), the CLIC project at CERN and to studies of a possible future 2.5 TeV linac. The program can be applied to a broad range of problems that vary widely in energy and beam parameters.

Interested scientists are explicitly invited to join the LIAR project and to contribute new features (commands). The LIAR code is put into the public domain and can be used and distributed freely. However, we expect that publications that contain LIAR results make proper reference to this user's guide. In addition we ask that any extensions and modifications to this program are made available to the scientific community for free usage. Contributions and questions should be submitted by e-mail to *assmann@slac.stanford.edu*.

2.1 Cross-checks

The name of this program does not indicate that all its results are wrong. Indeed, some effort was invested to make sure that LIAR provides correct answers. LIAR results were cross-checked against the following computer programs:

- Transport [1].
- LTrack [2].
- Turtle [3].
- Private codes from Raubenheimer, Adolphsen and Kubo [4].

The results show excellent consistency. In addition, SLC measurements were simulated with very good agreement. However, as always, there is no kind of warranty that results are correct. Bugs probably exist and LIAR might occasionally "lie". Problems should be reported by e-mail to *assmann@slac.stanford.edu*.

2.2 Compatibility

The LIAR code is mainly written in standard Fortran 77. It, however, takes advantage of the STRUCTURE and RECORD extensions that are available in most Fortran compilers. The code is stand-alone apart from a few system calls (random number generator, time) that need to be adjusted to the actual computer system. No specific libraries are required for the compilation. LIAR version 1.9 is presently running under those operating systems:

- UNIX: (IBM AIX 3.2, etc.).
  Computer: UNIX RISC workstations.
- Windows 95 / Windows NT: (Personal Computers).
  Computer: > Pentium 133 MHz, > 32 MB RAM.
  Compiler: Microsoft Powerstation Fortran 4.0.

The code is easily ported, as long as Fortran compilers are available that support the STRUCTURE and RECORD extensions. A Macintosh version can be generated using the appropriate version of the Microsoft Powerstation compiler. Unfortunately there is no free LINUX Fortran compiler available, that supports STRUCTURE's and RECORD's or Fortran 90 features. An adequate commercial compiler for LINUX is available from Absoft (compare http://marie.mit.edu/ templon/fortran.html).

2.3 Updates, bug fixes and additional information

The most recent information on LIAR is available through its home page on the World Wide Web

http://www.slac.stanford.edu/grp/arb/rwa/liar.htm

Please check for changes in the User's Manual or use the ONLINE manual with the most recent information.

New versions of LIAR will be put to its AFS site:

/afs/slac.stanford.edu/public/software/liar/release

Bug fixes will generally not result in a new version number. The existing version of LIAR will just be updated. Every 1-2 months, however, a new version will be released that contains all the old features plus the new commands that have been added since the last release. If existing commands are enhanced or significantly changed they will be available under a slightly changed name. The original commands with their old functionality will be available under their old name. We thus will try to maintain backward compatibility.

3 THEORY, CONCEPTS AND FEATURES

The overall design goal for LIAR was to provide a programming platform for simulations of linear accelerators. The basic structure was required to be flexible enough to include all beam physics and imperfections that are relevant to the operation and optical design of linear accelerators. At the same time it had to be easy enough to allow additions and extensions by non-experts. A working knowledge of FORTRAN should be sufficient to join the LIAR project and to help supplying new features. An interactive command language was implemented in order to ensure maximum ease of use. In the following we summarize the features and basic concepts of LIAR version 1.9. In order to describe the charged particle beam transport we use the TRANSPORT formalism [1]. Modifications are mainly necessary to include wakefield effects and the beam acceleration. LIAR profited from the experience with earlier simulation programs [4].

3.1 Phase space description

A position in phase space is defined by the 6-dimensional vector

$$X = \begin{pmatrix} x \\ x' \\ y \\ y' \\ z \\ E \end{pmatrix} \qquad \text{where } z < 0 \text{ is the head.} \tag{1}$$

The head of the bunch is located in the negative z direction. Here, x and x' are the horizontal displacement and angle, y and y' are the vertical displacement and angle, z is the longitudinal distance to the nominal center of the bunch and E is the total energy. Note that the definition of the energy coordinate differs from the TRANSPORT convention.

The vector X describes the centroid motion of a thin longitudinal slice of a bunch. With each slice we can also associate a beam matrix σ. Since the bunch length is constant in linear accelerators, it is sufficient to consider the beam matrix for the transverse planes only. It can be written as:

$$\sigma = \begin{bmatrix} \sigma_{xx} & \sigma_{xx'} \\ \sigma_{xx'} & \sigma_{x'x'} \end{bmatrix} = \epsilon \begin{bmatrix} \beta & -\alpha \\ -\alpha & \gamma \end{bmatrix} \tag{2}$$

Here β, α and $\gamma = (1+\alpha^2)/\beta$ are the Twiss parameters, related by the Courant-Snyder invariant:

$$\gamma x^2 + 2\alpha x x' + \beta x'^2 = \epsilon \tag{3}$$

This equation describes the *beam* ellipse which in a linear accelerator must be distinguished from the so-called *machine* ellipse (compare [5]). The lattice is designed for a given set of initial beam conditions. This determines the design or machine ellipse (matched case). If the beam is injected with different initial conditions then its Twiss values and the beam ellipse will be different (unmatched case). Even if the beam is injected with the initial conditions matched, imperfections will generally cause the beam ellipse to differ from the machine ellipse.

3.2 Beam description

The beam with a total charge of Q is described as a train of N_b bunches:

$$Q = \sum_{j=1}^{N_b} Q_j \tag{4}$$

Each bunch is longitudinally divided into N_s slices that are located at different positions z_i. Note that z_i is zero for the center of the bunch (without bending magnets) and negative for its head. Γ_z shall be the charge distribution along z (it might be Gaussian) and its integral shall be one. Γ_z is a function of the bunch length σ_z. The bunch charge Q_j is distributed into N_s slices at positions z_i as follows:

$$Q_j = \sum_{i=1}^{N_s} \Gamma_z(\sigma_z, z_i) \cdot Q_i \tag{5}$$

For a Gaussian distribution the slice positions z_i are determined from the bunch length σ_z, the number of slices N_s and the number N_σ of standard deviations to be considered on each side:

$$z_i = \left(i - \frac{N_s + 1}{2}\right) \cdot \frac{2N_\sigma \sigma_z}{N_s} \tag{6}$$

Finally each slice is divided into N_m mono-energetic beam ellipses. Γ_E shall be the energy distribution in a slice. The function Γ_E depends on the uncorrelated energy spread σ_E. With σ_E, N_m and the number N_σ of standard deviations to be considered on each side we obtain the energies E_m:

$$E_m = \left(m - \frac{N_m + 1}{2}\right) \cdot \frac{2N_\sigma \sigma_E}{N_m} \tag{7}$$

and the corresponding charge distribution for each slice:

$$Q_i = \sum_{i=1}^{N_m} \Gamma_E(\sigma_E, E_m) \cdot Q_m \tag{8}$$

The beam description in version 1.9 is equidistant in slice position z_i along the whole accelerator. The next version of LIAR will eliminate this requirement and will allow for changes of the bunch length in bending magnets.

3.3 Beamline elements

Beamline elements that are defined in LIAR include

1. quadrupole magnets,
2. dipole magnets (not complete in LIAR 1.9),
3. RF accelerating structures,
4. beam position monitors (BPM's),
5. horizontal and vertical dipole correctors and
6. marker points.

Dispersive effects are included for all elements. In a more general version of this program one would like to include synchrotron radiation effects and sextupoles.

All elements are described as thick lenses. The transport matrices, $\mathbf{R}$, that are used, are those defined in the TRANSPORT formalism [1]. They were modified only to include the transport through the preceding drift space for each element. The beam is always tracked through the combination of an element plus its preceding drift space. This allows to maximize computing speed. The transport matrix for each element is calculated separately for each beam ellipse of each slice of each bunch (excluding drifts). Such all orders of dispersion are included.

The accelerating RF structures are treated in a special way. They are described by at least two structure pieces with an interleaved transverse wakefield deflection:

$$\mathbf{R}_{\text{rf}} = \mathbf{R}_{\text{rf/2}} \cdot \mathbf{R}_{\text{WF}} \cdot \mathbf{R}_{\text{rf/2}} \tag{9}$$

The matrix $\mathbf{R}_{\text{rf/2}}$ transports the beam through one half of the structure. It takes into account the acceleration and longitudinal beam loading. The matrix $\mathbf{R}_{\text{WF}}$ describes the transverse wakefield deflections experienced by each slice of the beam. Transverse wakefield kicks are thus lumped together into the middle of the structures. This ansatz is refined by breaking up the structures into many pieces[2]. Each structure piece is then treated with the ansatz shown above. Tracking through the RF structures involves convoluting the beam with beam-excited transverse and longitudinal wakefields.

3.4 Beam energy and RF phases

A major problem in the operation and optimization of a linear accelerator is the choice and implementation of the right phases between the beam and the accelerating RF. Adjustments in the RF phases allow for example the implementation of bunch compressors or BNS damping. Changing the RF phases will modify the effective acceleration and the beam energy. It is necessary to readjust the beam energy to its design value and to scale the quadrupole strengths accordingly. Longitudinal wakefields will cause further energy loss that must be compensated in order to avoid a lattice mismatch. LIAR provides for a consistent way to adjust the RF phases and the final beam energy. The energy loss δE from single-bunch longitudinal wakefields is estimated in lattice calculations from

$$\delta E = -K_{\text{loss}} \cdot Q \cdot L \tag{10}$$

where K_{loss} is the loss factor (e.g. $K_{\text{loss}} = 5.05 \times 10^{14}$ V/C/m for NLC and $K_{\text{loss}} = 6.6 \times 10^{13}$ V/C/m for SLC), Q is the single bunch charge and L is the length of the RF structure[3].

3.5 Optics calculations

The Twiss values of the "machine ellipse" are calculated under the following assumptions:

1. zero bunch length
2. zero current
3. no transverse wakefield effects

However, longitudinal beam loading can be included with the simple analytical estimate from Eq. 10. The "machine" Twiss values are purely determined from the transport matrices of the beamline elements. The same matrices that are used for tracking, are used for the optics calculation. However, no particle tracking is done. We call the result of this calculation the "design model". This model is used for all correction algorithms where the transport matrix between two points in the accelerator needs to be known. Due to wakefields and imperfections, this model is only approximately correct. However, it is easily known and it is sufficient for the accelerator operation, as long as correction schemes are iterated. In order to improve

[2] In order to divide a structure into pieces, the lattice input files do not need to be changed. A command parameter in READ_TRNS and READ_MAD allows to break a structure into a number of pieces from within LIAR.

[3] The parameters for the short-range longitudinal wakefields are specified in the commands that make use of this simple approximation.

the convergence of correction methods, an improved "wakefield model" might be used. Serious correction problems will most likely result from local errors in the quadrupole fields, the acceleration and the RF phases. Those errors can be set in LIAR and can be included or excluded from the Twiss calculation.

3.6 Beam tracking

Beam ellipses are tracked fully coupled in 6D (without synchrotron radiation). Both the centroids X and the beam ellipses σ are tracked through the lattice:

$$X_1 = \mathbf{R}\, X_0 \tag{11}$$

and

$$\sigma_1 = \mathbf{R}\, \sigma_0\, \mathbf{R}^T \tag{12}$$

The beam can be defined as a sequence of electron and positron bunches. The lattice is scaled internally according to the sign of the bunch charge. Wakefields can be arbitrarily switched on and off for the tracking.

3.7 Dipole wakefields

Both single-bunch and multi-bunch dipole wakefield effects are included for the transverse plane. The wakefields are read in as parameterizations from design specific input files (compare section 8.2). In the transverse planes, imperfections of the long-range wakefields can be defined. As long as there are no imperfections, the transverse wakefield kick per unit offset per energy per charge is the same in all structures. Long-range wakefield imperfections are defined for a small number of "imperfection types" that are randomly assigned to the RF structures. Long-range longitudinal wakefields can be represented as well.

A slice i experiences an energy loss δE_i due to short-range longitudinal wakefields that are excited by all preceding slices j. It is determined as follows:

$$\delta E_i = \left[\frac{W_{\parallel}^{\mathrm{SR}}(0)}{2} \, |Q_i| + \sum_{j=1}^{i-1} W_{\parallel}^{\mathrm{SR}}(i-j) \cdot Q_j \right] \cdot L \tag{13}$$

Here the $Q_{i,j}$ are the charges in the slices i or j and $W_{\parallel}^{\mathrm{SR}}$ is the wakefield function as determined from the design specific input file. The head of the bunch is $i = 1$ and the sum term is only evaluated for $i > 1$. L is the length of the structure.

A slice l in a bunch k experiences a transverse trajectory deflection θ_x due to short-range and long-range wakefields that are excited by the preceding slices in all preceding bunches. The deflection angle is:

$$\theta_x(k,l) = \sum_{j=1}^{k-1} Q_j x_j L \left[W_{\perp}^{\mathrm{LR}}(k-j,P,T,E) + z_l \cdot W_{\perp}^{\prime\mathrm{LR}}(k-j,P,T,E) \right] \tag{14}$$

$$+ \sum_{i=1}^{l-1} Q_i x_i L W_{\perp}^{\mathrm{SR}}(l-i) \tag{15}$$

The Q_j are the bunch charges and the Q_i the slice charges. $W_{\perp}^{\mathrm{LR}}(k-j,P,T,E)$ is the long-range wakefield function for the bunch "distance" $(k - j \cdot \Delta s_{\mathrm{bunch}})$, the structure piece P, the structure type T and the error type E. $W_{\perp}^{\prime\mathrm{LR}}(k-i,P,T,E)$ is its derivative with respect to the longitudinal offset z_l to the bunch center. Long-range wakefields can be defined for different structure types T and different error types E. The term $W_{\perp}^{\mathrm{SR}}(l-i)$ specifies the short-range transverse wakefield function for a slice distance $(l - i \cdot \Delta s_{\mathrm{slice}})$.

3.8 Quadrupole wakefields

If a beam possesses a transverse quadrupole moment relative to the accelerator pipe axis, e.g., if the beam is not round or off axis, a quadrupole wake is generated. Although typically much weaker than the dipole wake, the quadrupole wake may not always be neglected, since it is not easily recognized and its effect can be orthogonal to that of the dipole wake. The implementation in LIAR closely follows the prescription by Chao and Cooper [11].

Two types of quadrupole moments exist: a normal quadrupole moment $Q_{q,1}$ and a skew quadrupole moment $Q_{q,2}$, which are defined by

$$Q_{q,1} = < x^2 > - < y^2 > \tag{16}$$
$$Q_{q,2} = 2 < xy > \tag{17}$$

The wake field generated by a slice with quadrupole moments $Q_{q,1}$ and $Q_{q,2}$ gives rise to a Lorentz force on another slice following a distance z behind it [11], which reads

$$e\left(\vec{E} + \frac{\vec{v}}{c} \times \vec{B}\right) = e^2 W(z)\left[Q_{q,1}(x\hat{x} - y\hat{y}) + Q_{q,2}(y\hat{x} + x\hat{y}\right] \tag{18}$$

where $W(z)$ is the quadrupole wake function, and $\hat{x}$ and $\hat{y}$ denote unit vectors in the horizontal and vertical direction.

In LIAR only the short-range quadrupole wake field is implemented. For the kth slice, the transfer matrix through a half structure of (half) length L is given by [11]

$$R_{quad} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ q_1 & 1 & q_2 & 0 \\ 0 & 0 & 1 & 0 \\ q_2 & 0 & -q_1 & 1 \end{bmatrix} \tag{19}$$

where

$$q_1 = \frac{r_e L}{\gamma_k} \sum_{i=1}^{k-1} N_i W_2(z_i - z_k) Q_{q,1,i} \tag{20}$$

and

$$q_2 = \frac{r_e L}{\gamma_k} \sum_{i=1}^{k-1} N_i W_2(z_i - z_k) Q_{q,2,i} \tag{21}$$

The sums are taken over all the previous slices, and N_i is the charge of slice i in units of e. The quadrupole moments of the ith slice are

$$Q_{q,1,i} = \sigma_{xx,i} - \sigma_{yy,i} + x_i^2 - y_i^2 \tag{22}$$

and

$$Q_{q,2,i} = 2\sigma_{xy,i} + 2x_i y_i \tag{23}$$

where the σ's denote elements of the sigma matrix for the ith slice. The quadrupole wake function for the SLAC linac was calculated by Bane and Wilson [11, 12]:

$$W_2(\Delta z) \approx 0.38 \times 10^{10}\ \mathrm{m}^{-5}\ (\Delta z/1.5\ \mathrm{mm})^{0.693-0.16(\Delta z/1\ \mathrm{mm})} \tag{24}$$

The quadrupole wake function for other accelerators can be estimated by scaling from the dipole wake function.

3.9 Static imperfections

Imperfections are understood to be static if their magnitudes do not significantly change on an hourly time scale. Here we give a list of error types that are presently implemented:

1. *Quadrupoles:* Transverse positions, rotation angle about longitudinal direction (roll), strength.

2. *BPM's:* Transverse positions, transverse positions with respect to quadrupole centers, finite resolution.

3. *Accelerating structures:* Transverse positions (offset misalignment), gradient, RF phase. Structures are misaligned on girders, as whole structures or in sub-pieces.

3.10 Dynamic imperfections

Dynamic imperfections are imperfections which may change from pulse to pulse or may drift significantly over minutes or hours.

1. *Ground motion:* Random ground motion (vibrations) and drifts as predicted from the ATL-model [9]:

$$\sigma^2 = A \cdot T \cdot L \tag{25}$$

   where T is the time in seconds and L is the distance in meters, and A is a constant.

2. *Timing fluctuations:* Random or systematic changes of the RF phase in the entire linac.

3. *Initial conditions:* Random and systematic changes of the initial beam parameters ("incoming position/intensity/phase jitter").

3.11 Correction schemes

Emittance preservation in accelerators requires the application of several correction algorithms. In LIAR we implemented a number of different methods.

3.11.1 1-to-1 correction

Standard correction scheme with dipole correctors. Two choices are available:

1. Least-square minimization of the BPM readings using dipole correctors over a range of the linac. The 1-to-1 approach means that exactly one downstream BPM is assigned to every dipole corrector. The number of BPM's is equal to the number of dipole correctors. Usually only correctors at high β values are used for trajectory correction. This means that the information of half of the BPM's is not used in the 1-to-1 correction scheme.

   For the least-square minimization the R_{12} elements of the beam transfer matrices are needed. If the dipole corrector is located at location 1 and the BPM at location 2 then the R_{12} is calculated from the Twiss parameters:

$$R_{12} = \sqrt{\frac{E_1}{E_2}} \sqrt{\beta_1 \beta_2} \cdot \sin(\Delta\psi) \tag{26}$$

where $\Delta\psi$ is the phase advance between the two points, E_1 and E_2 are the beam energies and β_1 and β_2 the β-functions. We then solve for the dipole corrector strengths K_i in order to minimize all BPM readings x_j:

$$\min\left[\begin{pmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{pmatrix} + \begin{pmatrix} R_{12}^{1\to 1} & 0 & \cdots & 0 \\ R_{12}^{1\to 2} & R_{12}^{2\to 2} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ R_{12}^{1\to 2} & R_{12}^{2\to 2} & \cdots & R_{12}^{n\to n} \end{pmatrix} \cdot \begin{pmatrix} K_1 \\ K_2 \\ \cdots \\ K_n \end{pmatrix}\right]^2 \tag{27}$$

The straight-forward Twiss calculation (compare above) does not include any wakefield effects. This 1-to-1 optimization algorithm should therefore be iterated in small regions in order to converge to the final solution.

2. Iterative minimization of each BPM reading by using one upstream dipole corrector ($\approx \pi/2$ upstream). This is a fast approach that works well in simulation but cannot easily be realized in large accelerators.

3.11.2 Beam-based alignment and steering

Beam-based alignment of quadrupoles and accelerating structures. This algorithm minimizes the BPM readings so that it also works as a steering algorithm.

The algorithm minimizes the quadrupole and structure BPM readings by first moving the quadrupoles, thereby steering the trajectory, and then moving the accelerator structures to align them along the beam trajectory. Because of imperfections in the accelerator model, a long linac is typically divided into many shorter regions of 50 to 100 quadrupoles and the algorithm is applied to each region individually. To obtain full correction, one usually has to iterate the correction multiple times. The simulations include the effect of finite BPM resolution (reading-to-reading jitter) and accelerator component misalignments.

The algorithm determines the quadrupole movements in an attempt to align the magnets in a straight line between the first and last quadrupoles of the region being considered; the first and last quadrupoles of the region are not moved by the algorithm. The beam is then launched along the beamline by adjusting either the initial conditions, for the first region of the linac, or by adjusting a single dipole corrector located at the first quadrupole for all subsequent regions; only a single dipole corrector is needed to join regions because the beam trajectory should be centered at the first quadrupole which is the last quadrupole of the preceding region. Finally, weights can be added for the BPM resolution and the quadrupole movements; the nominal values are the expected BPM resolution and the expected quadrupole misalignments with respect to adjacent magnets. These weights will limit the magnitude of the moves, constraining the trajectory to lie along the pre-determined axis which can be assumed to be set by the initial mechanical alignment.

In specific, the quadrupole alignment algorithm finds the least squares solution to the problem:

$$\begin{pmatrix} m_1 \\ \vdots \\ m_N \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \mathbf{R_1}\begin{pmatrix} q_2 \\ \vdots \\ q_{N-1} \\ x_1 \\ x_1' \end{pmatrix} \quad \text{or} \quad \mathbf{R_i}\begin{pmatrix} q_2 \\ \vdots \\ q_{N-1} \\ \theta_1 \end{pmatrix} \tag{28}$$

with a weighting vector given by

$$\begin{pmatrix} 1/\sigma_{bpm\,1} \\ \vdots \\ 1/\sigma_{bpm\,N} \\ 1/\sigma_{quad2} \\ \vdots \\ 1/\sigma_{quad\,N-1} \\ 1/\sigma_{init} \\ \beta/\sigma_{init} \end{pmatrix} \tag{29}$$

The measurement vector consists of N BPM measurements m_i followed by N zeros which are used to limit the quadrupole movements. The solution vector consists of $N-2$ quadrupole movements followed by x_1 and x'_1 which are the initial conditions or θ_1 which is a corrector located at the first quadrupole of the region. Next, the weighting vector consists of σ_{bpm}, σ_{quad}, and σ_{init} which are the estimated BPM resolution, quadrupole misalignments and initial error which nominally would be equal to the quadrupole misalignments. Finally, the matrix $\mathbf{R}$ is given by:

$$\mathbf{R_1} = \begin{pmatrix} 0 & 0 & \cdots & 0 & R_{11} & R_{21} \\ -1 & 0 & \cdots & 0 & R_{11} & R_{21} \\ K_2R_{12} & -1 & \cdots & 0 & R_{11} & R_{21} \\ K_2R_{12} & K_3R_{12} & \cdots & 0 & R_{11} & R_{21} \\ \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\ K_2R_{12} & K_3R_{12} & \cdots & K_{N-2}R12 & R_{11} & R_{21} \end{pmatrix} \tag{30}$$

where K_i is the integrated quadrupole strength, R_{12} is the $(1,2)$ transport matrix element from the i^{th} quadrupole to the BPM, R_{11} and R_{21} are the $(1,1)$ and $(2,1)$ matrix elements from the initial point to the BPM's.

After applying the quadrupole solution, the movers on the accelerator structures are adjusted such that the average RF-BPM reading on a girder is minimized. Each structure has two RF BPM's, one at either side. Let's assume that $\tilde{N}$ structures are mounted on a single girder that has a mover at either end. We then have $2 \cdot \tilde{N}$ RF BPM readings x_i at the locations s_i with i running from 1 to $\tilde{N}$. We then obtain the girder to beam offset Δx at every location s from

$$\Delta x(s) = m \cdot s + c \tag{31}$$

with

$$m = \frac{\sum_{i=1}^{2\cdot\tilde{N}} (s_i - \overline{s}) \cdot x_i}{\sum_{i=1}^{2\cdot\tilde{N}} (s_i - \overline{s})^2} \qquad \text{and} \qquad c = \overline{x} - (m \cdot \overline{s}) \tag{32}$$

This is a simple straight line fit to the RF BPM readings. $\overline{s}$ and $\overline{x}$ indicate the averages over all s_i and x_i. In order to align the RF structures, a mover at a location s_m is moved by $-\Delta x(s_m)$. Adjusting both movers on a girder will thus minimize the average RF BPM reading.

3.11.3 Dispersion-free steering

Quadrupole misalignments (but also RF structure misalignments) cause both trajectory deflections and centroid dispersion. A measurement of the centroid dispersion then allows to compensate misalignments effectively. The dispersion η_x in LIAR is defined as the change $\Delta x(s)$ in trajectory offset for a relative energy change $\Delta E/E$:

$$\Delta x(s) = \eta_x(s)\frac{\Delta E}{E} \approx \eta_x(s)\frac{K}{\Delta K} \quad . \tag{33}$$

In order to determine the dispersion, $\Delta E/E$ is assumed to be constant along s. Equivalent to changes in the beam energy E is a change in the lattice strength K (for quadrupoles and correctors; transverse structure deflections can be neglected). For practical reasons dispersion in linear accelerators is best measured with a lattice scaling.

Note that dispersion in a linear accelerator is not as uniquely defined as in a storage ring. However, the definition used here provides a dispersion that is a well defined observable. From the measured dispersion a dispersion-minimized trajectory can be calculated. This is called "Dispersion-free Steering" [13].

Since the superposition of many errors generates the dispersion, a model for how a deflection at any location changes the trajectory at all downstream locations is required. Neglecting wakefield effects, the absolute reading x^j at a BPM j due to all upstream kicks θ_i is

$$x^j = \sum_{i=1}^{j-1} R_{12}^{i\to j}\,\theta_i\,, \tag{34}$$

where the transport matrix elements $R_{12}^{i\to j}$ from the kick i to the BPM j are given by

$$R_{12}^{i\to j} = \sqrt{\frac{E^i}{E^j}}\cdot\sqrt{\beta_x^i\cdot\beta_x^j}\cdot\sin\left[\psi_x^i-\psi_x^j\right]\,. \tag{35}$$

Here the $E^{i,j}$ are the beam energies, the $\beta_x^{i,j}$ are the beta functions and the $\psi_x^{i,j}$ are the betatron phase advances.

Now we calculate how the dispersive kicks change the difference trajectory Δx^j when the lattice is scaled (equivalent to change in beam energy). For the scaled lattice we need to recalculate the Twiss parameters (primed quantities). Then obtain for Δx^j

$$\Delta x^j = x^j - x'^j = \sum_{i=1}^{j-1} R_{12,\kappa}^{i\to j}\,\theta_i\,, \tag{36}$$

where the $R_{12,\kappa}^{i\to j}$ are the transport matrix elements for the scaled lattice. We define a lattice scaling factor κ from the change ΔK in the quadrupole strength K as

$$\kappa = \frac{\Delta K}{K} + 1\,. \tag{37}$$

Effects from non-linear dispersion are neglected, since we use more than one lattice scalings for our analysis. The dispersion-free solution should be local. Dispersion bumps are not easily possible and linear and non-linear dispersion are minimized simultaneously. For a given lattice scaling κ the $R_{12,\kappa}^{i\to j}$ are

$$R_{12,\kappa}^{i\to j} = R_{12}^{i\to j} - \kappa\sqrt{\frac{E^i}{E^j}}\cdot\sqrt{\beta'^i_x\cdot\beta'^j_x}\cdot\sin\left[\psi'^i_x-\psi'^j_x\right]\,. \tag{38}$$

The Twiss parameters are calculated with the longitudinal magnet positions, the magnetic field values of the quadrupoles and the beam energy at each magnet.

The above model predicts the effect of dispersive deflections on the absolute trajectory x^j and the difference trajectories $\Delta x^j(\kappa)$. Alternatively, if we scale the lattice and measure x^j and $\Delta x^j(\kappa)$ we can use the model to calculate corrector settings that minimize both x^j and $\Delta x^j(\kappa)$. With four sets of measurements in each hysteresis cycle and n BPM's we define the vector B of measurements as

$$B = \begin{bmatrix} x^1 \\ \Delta x^1(\kappa_1) \\ \Delta x^1(\kappa_2) \\ \Delta x^1(\kappa_3) \\ x^2 \\ \Delta x^2(\kappa_1) \\ \Delta x^2(\kappa_2) \\ \Delta x^2(\kappa_3) \\ \vdots \\ x^n \\ \Delta x^n(\kappa_1) \\ \Delta x^n(\kappa_2) \\ \Delta x^n(\kappa_3) \end{bmatrix} , \quad W = \begin{bmatrix} W^1 \\ W^1_\Delta(\kappa_1) \\ W^1_\Delta(\kappa_2) \\ W^1_\Delta(\kappa_3) \\ W^2 \\ W^2_\Delta(\kappa_1) \\ W^2_\Delta(\kappa_2) \\ W^2_\Delta(\kappa_3) \\ \vdots \\ W^n \\ W^n_\Delta(\kappa_1) \\ W^n_\Delta(\kappa_2) \\ W^n_\Delta(\kappa_3) \end{bmatrix} , \tag{39}$$

with W as the vector of weights. The κ_i correspond to different lattice scale factors κ. For each of the n BPM's we have four measured quantities. The weights are defined through inverse measurement errors. Let's first consider the absolute trajectory measurement x^j. Its measurement error has a statistical contribution $\sigma(x^j) \approx \sigma_{\text{res}}$ and a systematic contribution σ_{bpm} from the absolute BPM misalignment. The statistical error $\sigma(x^j)$ arises mainly from the BPM resolution and is usually much smaller than the individual BPM misalignments. The weight on x^j is then

$$W^j = \frac{1}{\sigma^2_{\text{res}} + \sigma^2_{\text{bpm}}} . \tag{40}$$

Ideally the error on the measured trajectory difference Δx^j only has the statistical contributions of the two measurements. The weights on Δx^j are then defined by the BPM resolution σ_{res}

$$W^j_\Delta(\kappa_j) = \frac{1}{2\sigma^2_{\text{res}}} . \tag{41}$$

Equations 40 and 41 define the χ^2. The BPM resolution is usually much better than the BPM to quadrupole alignment and we can write approximately

$$\chi^2 \approx \sum_j \left[\frac{{x^j}^2}{\sigma^2_{\text{bpm}}} + \sum_{\kappa_i} \frac{{\Delta x^j}^2(\kappa_i)}{2\sigma^2_{\text{res}}} \right] . \tag{42}$$

Therefore, minimizing the dispersion makes optimum use of the BPM's and is much more efficient than solely minimizing the absolute trajectory. We next define a correlation matrix

$$A = \begin{bmatrix} R_{12}^{1\to1} & 0 & \ldots & 0 \\ R_{12,\kappa_1}^{1\to1} & 0 & \ldots & 0 \\ R_{12,\kappa_2}^{1\to1} & 0 & \ldots & 0 \\ R_{12,\kappa_3}^{1\to1} & 0 & \ldots & 0 \\ R_{12}^{1\to2} & R_{12}^{2\to2} & \ldots & 0 \\ R_{12,\kappa_1}^{1\to2} & R_{12,\kappa_1}^{2\to2} & \ldots & 0 \\ R_{12,\kappa_2}^{1\to2} & R_{12,\kappa_2}^{2\to2} & \ldots & 0 \\ R_{12,\kappa_3}^{1\to2} & R_{12,\kappa_3}^{2\to2} & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ R_{12}^{1\to n} & R_{12}^{2\to n} & \ldots & R_{12}^{n\to n} \\ R_{12,\kappa_1}^{1\to n} & R_{12,\kappa_1}^{2\to n} & \ldots & R_{12,\kappa_1}^{n\to n} \\ R_{12,\kappa_2}^{1\to n} & R_{12,\kappa_2}^{2\to n} & \ldots & R_{12,\kappa_2}^{n\to n} \\ R_{12,\kappa_3}^{1\to n} & R_{12,\kappa_3}^{2\to n} & \ldots & R_{12,\kappa_3}^{n\to n} \end{bmatrix} \tag{43}$$

and solve for the vector X of corrector settings:

$$\min_x ||W\ (\ B + AX)\ ||_2\ . \tag{44}$$

The solution X provides a set of corrector strengths that minimizes the trajectory and dispersion measurements simultaneously. Instead of solving for corrector kicks we could have solved for quadrupole positions that minimize the absolute trajectory and the dispersion.

3.11.4 Other beam-based correction schemes

Wake-free steering algorithm (not yet fully implemented).

3.11.5 Energy feedback

Fix the energy at a location using dedicated feedback klystrons (not yet implemented). The actual beam energy E is adjusted to the energy setpoint $E_{\rm setp}$. Note that we do not define any energy reference as for the position feedbacks.

3.11.6 Position feedback

A generic implementation without worrying about technical feasibility is realized. A feedback is defined by a reference position and angle

$$x_{\rm ref} \qquad \text{and} \qquad \text{x}'_{\rm ref} \tag{45}$$

and the setpoints

$$x_{\rm setp} \qquad \text{and} \qquad \text{x}'_{\rm setp} \tag{46}$$

at a point in the linac. Two upstream correctors that are reasonably close and efficient are used to adjust the actual beam position and angle ($x_{\rm beam}, x'_{\rm beam}$) to the reference values plus the setpoints:

$$x_{\rm beam} \quad \to \quad x_{\rm ref} + x_{\rm setp} \tag{47}$$

$$x'_{\rm beam} \quad \to \quad x'_{\rm ref} + x'_{\rm setp} \tag{48}$$

A feedback can be iterated in order to achieve the best agreement. In the LIAR implementation the $x_{\rm beam}$ and $x'_{\rm beam}$ at the feedback points are calculated with respect to the design plane. That is different from a real implementation, where a number of BPM readings are used to fit x_{beam} and x'_{beam} at one location in the linac. An obvious problem occurs when the beamline has large, long wavelength misalignments. A "real" feedback would not see those kinds of misalignments since the whole beamline locally moves. However, in LIAR the generic feedbacks would try to steer the beam back to the design plane which obviously is wrong. A command is provided to "misalign" the feedback reference to the beamline (FDBK_MISAL).

This generic feedback definition is well suited to describe the feedback stabilization of slow drifts for times larger than about 10 seconds. It is not suitable e.g. to study the pulse-to-pulse feedback response to fast changes.

3.11.7 Global emittance bumps

A global emittance bump correction is implemented. Emittance bumps are trajectory oscillations that are excited with a feedback at one point s_0 and closed with another feedback at some other point s_1. The minimal emittance using such a bump is obtained using a deterministic approach. Minimizing the emittance means minimizing the Courant-Snyder invariant. We define a χ^2 at an observation point downstream of the two feedbacks:

$$\chi^2 = \gamma\sigma_x^2 + 2\alpha\sigma_{xx'} + \beta\sigma_{x'}^2 \tag{49}$$

with

$$\begin{aligned}
\sigma_x^2 &= \sum_{i=1}^{N_s} W_i(x_i - y_i)^2 - \left\{\sum_{i=1}^{N_s} W_i(x_i - y_i)\right\}^2 && (50)\\
\sigma_{xx'}^2 &= \sum_{i=1}^{N_s} W_i(x_i - y_i)(x'_i - y'_i) - \left\{\sum_{i=1}^{N_s} W_i(x_i - y_i)\right\}\left\{\sum_{i=1}^{N_s} W_i(x'_i - y'_i)\right\} && (51)\\
\sigma_{x'}^2 &= \sum_{i=1}^{N_s} W_i(x'_i - y'_i)^2 - \left\{\sum_{i=1}^{N_s} W_i(x'_i - y'_i)\right\}^2 && (52)
\end{aligned}$$

The x_i and x'_i are the offset and divergence of every slice i before the application of the emittance bump. y_i and y'_i are the slice offsets/divergences needed to minimize the emittance. The number of the slices in the bunch is N_s. The weights W_i of the slices are defined as

$$W_i = \frac{Q_i}{Q} \tag{53}$$

where Q is the total bunch charge and Q_i is the charge of the slice.

The absolute minimum of the above χ^2 occurs when all the y_i, y'_i are equal to the x_i, x'_i. However, a single emittance bump cannot provide enough degrees of freedom to actually achieve that. We express the slice positions y_i and divergences y'_i in terms of the setpoints x_0 and x'_0 for the first feedback:

$$\begin{aligned}
y_i &= A_i x_0 + B_i x'_0 && (54)\\
y'_i &= C_i x_0 + D_i x'_0 && (55)
\end{aligned}$$

The coefficients A_i, B_i, C_i and D_i are obtained experimentally by varying the setpoints x_0 and x'_0 while observing the y_i and y'_i.

Solving for the minimum χ^2 provides the following solution for the optimal setpoints:

$$\begin{bmatrix} x_0^{\text{opt}} \\ x_0^{\prime\text{opt}} \end{bmatrix} = \mathbf{H}^{-1} \cdot K \tag{56}$$

The vector K depends on the slice offsets and divergences at the observation point and is defined as follows:

$$K_1 = \gamma \left(\sum_i W_i x_i A_i - \sum_i W_i x_i \sum_i W_i A_i \right) + \beta \left(\sum_i W_i x_i' C_i - \sum_i W_i x_i' \sum_i W_i C_i \right) \tag{57}$$

$$+\alpha \left(\sum_i W_i x_i C_i + \sum_i W_i x_i' A_i - \sum_i W_i x_i \sum_i W_i C_i - \sum_i W_i x_i' \sum_i W_i A_i \right) \tag{58}$$

$$K_2 = \gamma \left(\sum_i W_i x_i B_i - \sum_i W_i x_i \sum_i W_i B_i \right) + \beta \left(\sum_i W_i x_i' D_i - \sum_i W_i x_i' \sum_i W_i D_i \right) \tag{59}$$

$$+\alpha \left(\sum_i W_i x_i D_i + \sum_i W_i x_i' B_i - \sum_i W_i x_i \sum_i W_i D_i - \sum_i W_i x_i' \sum_i W_i B_i \right) \tag{60}$$

The 2×2 matrix $\mathbf{H}$ depends only on the transfer coefficients A_i, B_i, C_i and D_i:

$$H_{11} = \gamma \left\{ \sum_i W_i A_i^2 - \left(\sum_i W_i A_i \right)^2 \right\} + \beta \left\{ \sum_i W_i C_i^2 - \left(\sum_i W_i C_i \right)^2 \right\} \tag{61}$$

$$+2\alpha \left\{ \sum_i W_i A_i C_i - \sum_i W_i A_i \sum_i W_i C_i \right\} \tag{62}$$

$$H_{22} = \gamma \left\{ \sum_i W_i B_i^2 - \left(\sum_i W_i B_i \right)^2 \right\} + \beta \left\{ \sum_i W_i D_i^2 - \left(\sum_i W_i D_i \right)^2 \right\} \tag{63}$$

$$+2\alpha \left\{ \sum_i W_i B_i D_i - \sum_i W_i B_i \sum_i W_i D_i \right\} \tag{64}$$

$$H_{12} = H_{21} = \gamma \left\{ \sum_i W_i A_i B_i - \sum_i W_i A_i \sum_i W_i B_i \right\} \tag{65}$$

$$+\beta \left\{ \sum_i W_i C_i D_i - \sum_i W_i C_i \sum_i W_i D_i \right\} \tag{66}$$

$$+\alpha \left\{ \sum_i W_i B_i C_i - \sum_i W_i B_i \sum_i W_i C_i + \sum_i W_i D_i A_i - \sum_i W_i D_i \sum_i W_i A_i \right\} \tag{67}$$

From equation 56 we obtain the following explicit expressions for the optimal setpoints:

$$x_0^{\mathrm{opt}} = \frac{1}{H_{11}H_{22} - H_{12}^2} \left(H_{22}K_1 - H_{12}K_2 \right) \tag{68}$$

$$x_0^{\prime\mathrm{opt}} = \frac{1}{H_{11}H_{22} - H_{12}^2} \left(H_{11}K_2 - H_{12}K_1 \right) \tag{69}$$

$$\tag{70}$$

Due to limitations of the beam instrumentation, the slice offsets and divergences are not known individually in a real accelerator. The bump setpoints are therefore optimized manually in order to minimize the measured bunch emittance. However, future advances in the beam instrumentation could provide for a deterministic emittance optimization like it is described here.

3.12 Observables

There are many relevant observables defined in LIAR. Obvious observables are the phase space positions at MARKERS (and part of it at BPM's) for slices, whole bunches or the whole beam. Also obvious are the "machine" Twiss parameters (β, α, ψ ...) at all elements and for electrons and positrons. We define the beam emittance with respect to the beam centroid as:

$$\epsilon = \left[\sigma_{xx}\sigma_{x'x'} - \sigma_{xx'}^2\right]^{\frac{1}{2}} \tag{71}$$

where

$$\sigma_{xx} = \frac{1}{Q}\sum_{i=N_0}^{N} Q_i\left[(x_i - \overline{x})^2 + \sigma_{xx,i}\right] \tag{72}$$

$$\sigma_{xx'} = \frac{1}{Q}\sum_{i=N_0}^{N} Q_i\left[(x_i - \overline{x})(x'_i - \overline{x'} + \sigma_{xx',i}\right] \tag{73}$$

$$\sigma_{x'x'} = \frac{1}{Q}\sum_{i=N_0}^{N} Q_i\left[(x'_i - \overline{x'})^2 + \sigma_{x'x',i}\right] \tag{74}$$

$$\overline{x} = \frac{1}{Q}\sum_{i=N_0}^{N} Q_i x_i \tag{75}$$

$$\overline{x'} = \frac{1}{Q}\sum_{i=N_0}^{N} Q_i x'_i \tag{76}$$

$$Q = \sum_{i=N_0}^{N} Q_i \tag{77}$$

The sum runs over the individual beam ellipses in the beam. N_0 and N can be chosen such that the emittance is calculated for the whole beam, a single bunch or a single slice. If the beam emittance is calculated, then N_0 is equal to 1 and N is equal to the product of the number of bunches, the number of slices per bunch and the number of mono-energetic beam ellipses per slice:

$$N_t = N_b \cdot N_s \cdot N_m \tag{78}$$

Q_i is the charge in the beam ellipse i and $\sigma_{xx,i}$, $\sigma_{x'x',i}$ and $\sigma_{xx',i}$ are the beam ellipse parameters as introduced in equation 2.

The emittance is not a measure of luminosity reduction. Large emittances often result from large tails that contain only little charge. A figure of merit for luminosity reduction can be obtained from a cross-correlation of a bunch with itself. Beam tails are naturally de-weighted. We define the following luminosity reduction factor R:

$$R = \frac{\overline{\sigma_x}}{Q^2}\sum_{j=1}^{N_s}\sum_{i=1}^{N_s}\frac{Q_iQ_j}{\left(\sigma_i^2 + \sigma_j^2\right)^{1/2}}\exp\left[-\frac{1}{2}\frac{(\overline{x_i} - \overline{x_j})^2}{\sigma_i^2 + \sigma_j^2}\right] \tag{79}$$

The double sum runs over all slices in a bunch. The $\overline{x_{i,j}}$ are the average centroid offsets of the slices and the $\sigma_{i,j}$ are the RMS sizes of the given slice. Note that this definition of the luminosity reduction factor oscillates. The final value obtained in LIAR is therefore averaged over the last 5% of the BPM's in the linac.

The mismatch β_{mag} between the machine ellipse and the beam ellipse is of great importance in linear accelerators. We use the following definition:

$$\beta_{\text{mag}} = \frac{1}{2}\left(\frac{\beta}{\beta^*} + \frac{\beta^*}{\beta}\right) + \frac{1}{2}\left(\alpha^*\sqrt{\frac{\beta}{\beta^*}} - \alpha\sqrt{\frac{\beta^*}{\beta}}\right)^2 \tag{80}$$

Here, α and β are the machine and α^* and β^* are the beam Twiss values. In the ideal matched case β_{mag} is equal to one.

3.13 Matching

Optical matching routines are not implemented at this time.

3.14 Trajectory fit

It is often necessary to fit the beam centroid offset and angle at some location "0" from a number "n" of downstream BPM measurements. This fit has a unique solution and $(x(0), x'(0))$ can be calculated analytically as:

$$x(0) = \left[\frac{\sum_{i=1}^{n} R_{12}(i) \cdot x(i)}{\sum_{i=1}^{n} R_{11}(i) \cdot R_{12}(i)} - \frac{\left[\sum_{i=1}^{n} R_{12}^2(i)\right] \cdot \left[\sum_{i=1}^{n} R_{11}(i) \cdot x(i)\right]}{\left[\sum_{i=1}^{n} R_{11}(i) R_{12}(i)\right]^2}\right] / \tag{81}$$

$$\left[1 - \frac{\left[\sum_{i=1}^{n} R_{11}^2(i)\right] \cdot \left[\sum_{i=1}^{n} R_{12}^2(i)\right]}{\left[\sum_{i=1}^{n} R_{11}(i) R_{12}(i)\right]^2}\right] \tag{82}$$

$$x'(0) = \frac{\sum_{i=1}^{n} R_{11}(i) \cdot x(i) - \left[\sum_{i=1}^{n} R_{11}^2(i)\right] \cdot x(0)}{\left[\sum_{i=1}^{n} R_{11}(i) R_{12}(i)\right]^2} \tag{83}$$

The fit naturally requires the knowledge of the R-matrix from the point to be fitted to all downstream BPM locations and the BPM readings $x(i)$. This fit is implemented in the command TRAJFIT.

3.15 Diagnostic linac model

The transfer matrices in a linac can be very different from the ones that are calculated in a single-particle model within LIAR. Reasons for this can be wakefield effects that are expected from the design (multi-particle beam dynamics) or errors in the beam energy, quadrupole strengths, etc. Analytical estimates for multi-particle beam behavior generally do not have the accuracy as required for the characterization of wakefield dominated linacs. In addition, parameter errors are by definition unknown in real accelerators. LIAR therefore implements beam diagnostic observables that are determined from "measured" BPM values. The same diagnostic method was applied for SLC very successfully. It allows to experimentally determine the "real" linac optics. We follow the method that was first described in [14].

The linac beam transport is best described by the R-matrices for the centroid motion. As an example, let's define the linac $R_{12,l}$'s. They can be written in a general form for locations "0" and "i" as:

$$R_{12,l} = \sqrt{\frac{E_l(0)}{E_l(i)}} \cdot \kappa_l \cdot \sqrt{\beta_l(0)\beta_l(i)} \cdot \sin(\Delta\psi_l) \tag{84}$$

This is a simple extension of Eq. 26 that was defined for design values without wakefields. The additional factor κ_l describes any blowup of an induced linac betatron oscillation (for example due to head to tail wakefield amplification). Without wakefields and errors κ_l is exactly one. Note that the centroid phase advance $\Delta\psi_l$ between two points in the linac can be strongly modified by wakefields, as well as the centroid β-functions.

The centroid beam transport in a linac can be described by examining the betatron oscillations (x_1, x_1') and (x_2, x_2') due to two beam deflections that are about π apart in phase advance (indicated by the subscripts 1 and 2). The R-matrix from an initial point number "0" (close to the origin of the beam deflections) to a downstream location number "i" is obtained from solving simultaneously:

$$\begin{pmatrix} x_1(i) \\ x_1'(i) \end{pmatrix} = \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix} \cdot \begin{pmatrix} x_1(0) \\ x_1'(0) \end{pmatrix} \tag{85}$$

$$\begin{pmatrix} x_2(i) \\ x_2'(i) \end{pmatrix} = \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix} \cdot \begin{pmatrix} x_2(0) \\ x_2'(0) \end{pmatrix} \tag{86}$$

The result is:

$$R_{11} = \frac{x_1(i) \cdot x_2'(0) - x_2(i) \cdot x_1'(0)}{x_1(0) \cdot x_2'(0) - x_2(0) \cdot x_1'(0)} \quad (87)$$

$$R_{12} = \frac{x_1(0) \cdot x_2(i) - x_1(i) \cdot x_2(0)}{x_1(0) \cdot x_2'(0) - x_2(0) \cdot x_1'(0)} \quad (88)$$

$$R_{21} = \frac{x_1'(i) \cdot x_2'(0) - x_2'(i) \cdot x_1'(0)}{x_1(0) \cdot x_2'(0) - x_2(0) \cdot x_1'(0)} \quad (89)$$

$$R_{22} = \frac{x_2'(i) \cdot x_1(0) - x_1'(i) \cdot x_2(0)}{x_1(0) \cdot x_2'(0) - x_2(0) \cdot x_1'(0)} \quad (90)$$

$$\quad (91)$$

The R-matrix is then transformed into normalized coordinates:

$$R'_{11} = \sqrt{\frac{\beta_0}{\beta}} R_{11} - \frac{\alpha_0}{\sqrt{\beta\beta_0}} R_{12} \quad (92)$$

$$R'_{12} = \frac{1}{\sqrt{\beta\beta_0}} R_{12} \quad (93)$$

$$R'_{21} = \alpha\sqrt{\frac{\beta_0}{\beta}} R_{11} - \frac{\alpha\alpha_0}{\sqrt{\beta\beta_0}} R_{12} + \sqrt{\beta\beta_0} R_{21} - \alpha_0\sqrt{\frac{\beta}{\beta_0}} R_{22} \quad (94)$$

$$R'_{22} = \frac{\alpha}{\sqrt{\beta\beta_0}} R_{12} + \sqrt{\frac{\beta}{\beta_0}} R_{22} \quad (95)$$

This transformation requires the knowledge of the optical functions α and β along the linac. Assuming that they are locally correct we use the design Twiss values from the single particle model.

Now we can define a number of observables that describe the “real” linac optics[4]:

1. Centroid linac phase advance ψ_l:

$$\psi_l = \text{arctan}\left(\frac{R'_{12}}{R'_{11}}\right) \quad (96)$$

2. Enhancement κ_l of linac oscillation amplitude:

$$\kappa_l = \frac{\sqrt{\det R'}}{\sqrt{E_0/E}} \quad (97)$$

3. Linac β-function:

$$\beta_l = \frac{(cR'_{11} + sR'_{12})^2}{\det R'} \cdot \beta \quad (98)$$

where c and s are defined as:

$$c = \cos\left[\text{arctan}\left(\frac{R'_{12}}{R'_{11}}\right)\right] \qquad \text{and} \qquad s = \sin\left[\text{arctan}\left(\frac{R'_{12}}{R'_{11}}\right)\right] \quad (99)$$

[4]Note that though we write the “arctan” function the “arctan2” function is used for the LIAR implementation.

4. Linac α-function:

$$\alpha_l = \frac{\left(cR'_{11} + sR'_{12}\right)\left(cR'_{21} + sR'_{22}\right)}{\det R'} + \frac{\beta_l}{\beta}\alpha \tag{100}$$

   where c and s are defined as above.

5. The linac β_{mag} mismatch is obtained with Equation 80, the linac optical functions β_l and α_l and the design optical functions β and α.

In order to describe a wakefield dominated linac completely, the two beam deflections must be induced along the whole length of the linac, resulting in a two-dimensional table of R_{12}'s. In LIAR we have implemented the simulation and analysis of two incoming betatron oscillations as obtained from a possible diagnostic pulse (see command MEAS_PHASE2).

4 GETTING STARTED WITH LIAR

This section provides a basic introduction to LIAR. First we explain how to set up the program. Then we describe the basic rules of the LIAR command language and the possibility of automatic command loops. The standard LIAR output is explained and we provide two examples of both simple and advanced SLC simulations.

4.1 Setup

LIAR is presently implemented for UNIX and MS windows operating systems. Its home directory is located in a SLAC AFS account under UNIX. The released version is world-readable and is located at:

/afs/slac.stanford.edu/public/software/liar/release

The directory tree is as follows:

.../bin.aix6000 Executable for the IBM AIX unix system.

.../examples Example command files.

.../doc Documentation in LaTex and Postscript format.

.../infiles Required input files (lattices,...).

.../run Directory to run command examples.

.../src_v1.9 Source code for LIAR version 1.9 (including Makefile).

SLAC workstations that are suitable for running LIAR include the SCS public machines *morgan01* to *morgan06*. The *liar* directory tree can be copied to a private UNIX machine and recompiled using the *Makefile* which is provided. A VAX-VMS version is foreseen but is not yet compiled (volunteers are welcome).

In order to run LIAR as a SLAC user, get your personal UNIX account and e.g. login on *morgan01*. Then you could proceed:

1. Copy directory tree

   ```
   > cp -r -p /afs/slac.stanford.edu/public/software/liar/release liar
   ```

2. Change to run directory

   ```
   > cd liar/run
   ```

3. Copy example command file

   ```
   > cp ../examples/cmdslc1 .
   ```

4. Run LIAR and execute command file...

   ```
   > liar cmdslc1
   ```

5. ...or execute LIAR in interactive mode

```
> liar
```

The Windows 95 PC version of LIAR runs as a console application in a DOS window.

LIAR can be run in two different modes. If it is started from the command level without any parameters (e.g. *liar*) then it will enter into an interactive mode and the user is prompted for input:

```
         ___        ___
         |||        |||
         |||        ___     ______     _____
         |||        |||   _||||||    _||||
         |||        |||   |||  ||    |||
         |||____    |||   |||__||_   |||
         |||||||    |||   ||||||||   |||    Version 1.9
         ________________________________________________
         L I near    A ccelerator    R esearch   code

         S L A C    1997    ---    Ralph Assmann
                                   Karl Bane
                                   Tor Raubenheimer
                                   Kathy Thompson


LIAR>
```

By typing the command

```
LIAR>  load, file='cmdfile'
```

the command file *cmdfile* would be executed from the interactive program level. After all commands in the file have been executed the control is returned to the user who could now submit a few final commands to complete the job. The interactive mode is used best for small simulations or for making sure that a bigger job will provide useful results. The *load* command can only be used from the interactive level of LIAR. Nesting of command files is not allowed.

LIAR automatically enters into a batch mode if it is started with a command file parameter. The UNIX command

```
> liar cmdfile
```

will start the program *liar* with the command file *cmdfile* as input. After all commands have been executed LIAR is stopped. The batch mode is highly recommended for large simulation jobs. Since LIAR uses about 35 MBytes of random access memory the use of system resources is minimized by running in batch mode.

4.2 Introduction to the command language

The LIAR command language is inspired from the MAD style [6]. However, our implementation is significantly different, allowing several flexible extensions. At the same time some restrictions are introduced. Certain rules must be followed for the command line syntax. All those rules apply to both the interactive and the batch mode of LIAR.

1. A command line (that can extend over several lines) consists of a *command name* and any number of *command options*:

```
LIAR>  track,    order=1,  couple = .f.
         |         |          |
       command   option     option
```

Command options can be specified in any order and many of them can be omitted. Parameters that are omitted use their default values. LIAR is not case sensitive[5].

2. All parts of the command line must be separated by *commata*. Spaces in the command line are ignored.

3. An input line that is ended by a comma is automatically continued into the next line. A command is ended by breaking a line without a comma.

```
LIAR>  track, order  = 1,
_             couple = .f.
```

The underscore '_' is added automatically by the command line parser.

4. Command options consist of the option *name* and its *value*. The values are assigned with an equal sign. It is essential that the values are specified in the following way:

   (a) Logicals are specified by .t., .T., .f. or .F. (e.g. “couple=.t.”)! If a parameter is specified without an equal ('=') sign, then it is assumed to be of type logical and to have the value .T.! So specifying “couple” is equivalent to “couple=.t.”.
   (b) Character strings must be enclosed in SINGLE quotes (e.g., file = 'run/TEST'). Note that UNIX filenames are case sensitive, though LIAR is not.
   (c) Every parameter that is not recognized as a logical variable or a character string and that contains an equal ('=') sign, is read in as a REAL value. Therefore failing to obey to the rules described will result in errors.

5. Each input line can take a maximum of 80 characters. A single command is allowed to take up to 10 input lines (9 continuation lines).

6. Comments are indicated by '!'. Everything which follows on the particular input line is disregarded.

7. The use of TAB's is not allowed. It will result in errors.

This is an example of a valid command line syntax:

```
 ERROR_GAUSS_QUAD, X_SIGMA = 5.E-06,     ! comment
                   Y_SIGMA = 7.E-06,     ! comment
                   T_SIGMA = 1.E-06,     ! comment
                   NAME = 'Q02*'         ! comment
```

Lines of comments within the scope of a single command are not allowed. The following command would fail:

```
  ERROR_GAUSS_QUAD, X_SIGMA = 5.E-06,     ! comment
!                    Y_SIGMA = 7.E-06,     ! comment (WRONG)
                    T_SIGMA = 1.E-06,     ! comment
                    NAME = 'Q02*'         ! comment
```

[5] However, keep in mind that UNIX filenames are case sensitive!

4.3 Iterative command loops

An advanced feature of the LIAR command language is the possibility to specify iterative loops over groups of commands. Up to three nested levels of do loops are allowed. For each loop a step variable is defined that can be used to specify command options. The basic do loop is implemented in a FORTRAN fashion:

```
DO SDY = 1E-6, 10E-6, 10
  ...
  ERROR_GAUSS_QUAD, X_SIGMA = 5.E-06,     ! comment
                    Y_SIGMA = @SDY,       ! comment
                    T_SIGMA = 1.E-06,     ! comment
                    NAME = 'Q02*'         ! comment
  ...
END DO
```

In this example the variable SDY is stepped up from 1×10^{-6} to 10×10^{-6} in 10 steps. In the misalignment command its value is accessed by @SDY. The implementation of do loops allows also to vary the seed of misalignments or to modify filenames for the output (e.g. FILE='output@SEED'). Using DO variables for output filenames will only take into account the integer part of the variable (the allowable range is 0 to 1000).

Do loops are only permitted in input command files (executed with the commands LOAD, EXEC or READ). They cannot be specified at the LIAR prompt. Do loops are expanded during a special preprocessing of an input file (e.g. 'input.liar'). If anything goes wrong the expanded input files (e.g. 'input.liar_1', 'input.liar_2' and 'input.liar_3') can be found in the actual directory and should be checked for errors. The expanded file with the highest number will be actually executed. It should not contain any 'DO' or 'ENDDO' commands!

4.4 Standard LIAR output

LIAR provides extensive information during the execution of its commands. For example the progress of the tracking is indicated by a "trackometer":

```
T R A C K O M E T E R :

0    10    20    30    40    50    60    70    80    90    100 %
|____|____|____|____|____|____|____|____|____|____|
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

At the end of the tracking, summary information is printed to the standard output:

```
     End of tracking   A N A L Y S I S :

     BEAM ENERGY
        - acceleration:     E_0    =    1.190 GeV   -->   E_f    =   46.003 GeV
        - spread:           E_sig  =     .016 GeV   -->   E_sig  =     .124 GeV
        - rel. spread      SIGE/E  =    1.322 %     -->  SIGE/E  =     .269 %
     BEAM BLOW-UP
        - Emittance (b1):   g_x    =   31.621 %           g_y    =   41.845 %
                    (av):   g_x    =   31.621 %           g_y    =   41.845 %
                 (train):   g_x    =   31.621 %           g_y    =   41.845 %
        - Lumin. factor:    R_x    =     94.6 %           R_y    =     95.1 %
        - BMAG mismatch:    BMAGX  =    1.030             BMAGY  =    1.212
     INITIAL BEAM SIZE:
        - Horizontal:       S_x    =    274.7 um          S_x'   =     64.6 urad
        - Vertical:         S_y    =     85.5 um          S_y'   =     24.0 urad
     FINAL BEAM SIZE:
        - Horizontal:       S_x    =  129.028 um          S_x'   =    3.685 urad
        - Vertical:         S_y    =   45.858 um          S_y'   =    1.247 urad
     JITTER OFFSETS WRT AXIS FOR BUNCH   0:
        - Horizontal:       J_x    =      .94 um          J_x'   =      .22 urad
        - Vertical:         J_y    =     2.11 um          J_y'   =     1.12 urad
     BEAM TRAJECTORY (IDEAL)
        - average:          X_av   =     -.10 um          Y_av   =     1.93 um
        - RMS:              X_sig  =   159.54 um          Y_sig  =   113.64 um
     BEAM TRAJECTORY (BPM READINGS)
        - average:          X_av   =      .02 um          Y_av   =    -3.67 um
        - RMS:              X_sig  =   150.00 um          Y_sig  =   110.48 um
     FINAL BETA FUNCTIONS
        - from beam:        B_x    =    27.64 m           B_y    =    51.34 m
        - from TWISS:       B_x    =    22.13 m           B_y    =    27.25 m
     FINAL ALPHA FUNCTIONS
        - from beam:        A_x    =    .7404             A_y    =   -1.403
        - from TWISS:       A_x    =    .7051             A_y    =    -.700
     FINAL GEOM. EMITTANCE
        - initial value:    E_x    = 1.29E-08 rad*m       E_y    = 1.50E-09 rad*m
        - wrt centroid:     E_x    = 4.39E-10 rad*m       E_y    = 5.51E-11 rad*m
        - wrt axis:         E_x    = 1.68E-09 rad*m       E_y    = 2.83E-10 rad*m
     FINAL NORM. EMITTANCE
        - initial value:    E_x    = 3.00E-05 rad*m       E_y    = 3.50E-06 rad*m
        - wrt centroid:     E_x    = 3.95E-05 rad*m       E_y    = 4.96E-06 rad*m
        - wrt axis:         E_x    = 1.52E-04 rad*m       E_y    = 2.55E-05 rad*m
```

This output information is explained in Table I. In addition most of the results are saved at every BPM location. They can be printed out into files, for example in order to study the emittance growth along the linac (compare summary descriptions in sections 5.13 and 5.14).

Headline	Label	Description
Beam energy	E_0	Initial beam energy in GeV.
	E_f	Final beam energy in GeV.
	E_sig	Absolute energy spread in GeV (start and end).
	SIGE/E	Relative energy spread in % (start and end).
Beam blow-up	g_x, g_y (b1)	X, Y emittance growth $\Delta\epsilon/\epsilon_0$ of the first bunch.
	g_x, g_y (av)	Average single bunch X, Y emittance growth $\Delta\epsilon/\epsilon_0$.
	g_x, g_y (train)	X, Y emittance growth $\Delta\epsilon/\epsilon_0$ for bunch train (incl. bunch offsets).
	R_x, R_y	X, Y luminosity estimate in % of ideal luminosity.
	BMAGX, BMAGY	X, Y beta mismatch (1 = no mismatch).
Initial beam size	S_x, S_y	Initial X, Y 1σ beam sizes.
	S_x' , S_y'	Initial X, Y 1σ beam divergence.
Final beam size	S_x, S_y	Final X, Y 1σ beam sizes.
	S_x' , S_y'	Final X, Y 1σ beam divergence.
Jitter offsets	J_x, J_y	Beam offsets with respect to the axis or a reference.
Beam trajectory (ideal)	X_av, Y_av	Average X, Y beam offset at BPM locations with respect to ideal reference line.
	X_sig, Y_sig	RMS X, Y beam offset at BPM locations with respect to ideal reference line.
Beam trajectory (BPM)	X_av, Y_av	Average X, Y beam offset with respect to BPM centers.
	X_sig, Y_sig	RMS X, Y beam offset with respect to BPM centers.
Final beta functions	B_x, B_y	X and Y beta function at the end of the machine from (a) tracked beam and (b) lattice Twiss calculation.
Final alpha functions	A_x, A_y	X and Y alpha function at the end of the machine from (a) tracked beam and (b) lattice Twiss calculation.
Geometric emittance	E_x, E_y	Geometric emittances at start and end of tracking. The final emittance is specified both with respect to the beam centroid and with respect to the ideal reference line.
Normalized emittance	E_x, E_y	Normalized emittances at start and end of tracking. The final emittance is specified both with respect to the beam centroid and with respect to the ideal reference line.

Table I: Definition of quantities that are printed out as the summary information after each tracking in LIAR.

4.5 MATHEMATICA interface

LIAR has been run within the MATHEMATICA environment. The interface allows to edit LIAR input files, run the program and to process the LIAR output files all within the MATHEMATICA environment. The mathematical power of MATHEMATICA can be very helpful in testing advanced optimization algorithms or in comparing analytical theories with simulation results.

The MATHEMATICA interface consists primarily in a number of functions that transfer LIAR information into MATHEMATICA objects and often also allow to plot them easily. Further information can be obtained from Gennadi Stupakov at SLAC (e-mail: stupakov@slac.stanford.edu).

4.6 Examples of SLC command files

We show two examples for SLC simulations. A simple SLC case allows the new user to get familiar with the concept of the LIAR command language and the basic philosophy in setting up a simulation. An advanced simulation demonstrates the capabilities of the computer program in defining multiple imperfections and advanced correction schemes.

4.6.1 Simple SLC study

The first example considers a simple case where we just simulate the effect of an injection offset of 200 μm in both planes. It includes wakefield and dispersion effects and a setup for BNS damping. The trajectories and emittances are calculated. The input file can be found in

```
/afs/slac.stanford.edu/public/software/liar/release/examples/cmdslc1
```

The listed commands can be entered interactively or read in as a command file by specifying

```
LOAD, FILE = 'filename'
```

at the LIAR prompt (alternatively the READ or EXEC commands can be used). Note that exclamation marks ('!') indicate comments. In this example the command names are printed in upper case, all options in lower case. Note however that LIAR is not case sensitive (though UNIX file names are). We first show the complete command input file. We then will go through the details of the command file, explaining every step.

```
  RESET, all
!
  READ_TRNS, infile  = '../infiles/slc.trns',
             energy  = 1.19,
             npiece  = 1
!
  SET_RF, energy   = 46.,
          scale    = .t.,
          kloss    = 0.66d14,
          charge   = 3.5d10,
          phase1   = 22.,
          lswitch1 = 800.,
          phase2   = -16.5
!
  CALC_TWISS, betax   = 3.40,
              betay   = 3.08,
              alphax  = 0.156,
              alphay  = 0.066
!
```

```
  SET_INITIAL, x        = 200.d-6,
               y        = 200.d-6
               energy   = 1.19,
               espread  = 0.014,
               nemitx   = 3.d-5,
               nemity   = 0.35d-5,
               match
!
  SET_BEAM, current  = 3.5d10,
            blength  = 1100.d-6,
            nb       = 1,
            ns       = 20,
            nm       = 3
!
  SET_SR_WF, file    = '../infiles/srwf_slc.dat'
!
  TRACK
!
  MEAS_BPM,  file = 'bpm.dat'
```

Now we step through this input file and explain the commands and their options. Note that we do only specify the necessary options. Many more options exist, but we use their default values.

In the first simulation step all data areas are initialized to zero:

```
RESET, all
```

Now we read in an SLC transport deck, define the injection energy and tell LIAR to treat each structure in a single piece:

```
READ_TRNS, infile  = '../infiles/slc.trns',
           energy  = 1.19,
           npiece  = 1
```

The RF is set up with the command SET_RF. We specify the final beam energy ('energy'), a first RF-phase ('phase1'), a switching point ('lswitch1') for the RF-phase at 800 m and a second RF-phase. The lattice and the acceleration is scaled ('scale') to fit those RF-phases and the final beam energy. We also compensate for longitudinal beam loading with a simple model ('kloss' and 'charge').

```
SET_RF, energy   = 46.,
        scale    = .t.,
        kloss    = 0.66d14,
        charge   = 3.5d10,
        phase1   = 22.,
        lswitch1 = 800.,
        phase2   = -16.5
```

It is always a good idea to calculate the Twiss values at all elements, though not needed for our example. The initial Twiss values must be specified:

```
CALC_TWISS, betax   = 3.40,
            betay   = 3.08,
            alphax  = 0.156,
            alphay  = 0.066
```

We specify the injection energy, the initial energy spread and the initial normalized emittances. The 'match' option tells LIAR to match the beam to the Twiss values that were specified before[6]. We also define an horizontal and vertical injection offset of 200 μm ('x' and 'y'):

```
SET_INITIAL, x        = 200.d-6,
             y        = 200.d-6
             energy   = 1.19,
             espread  = 0.014,
             nemitx   = 3.d-5,
             nemity   = 0.35d-5,
             match
```

Next we define the bunch population, the bunch length, the number of bunches ('nb'), the number of slices per bunch ('ns') and the number of mono-energetic beam ellipses per slice ('nm'):

```
SET_BEAM, current  = 3.5d10,
          blength  = 1100.d-6,
          nb       = 1,
          ns       = 20,
          nm       = 3
```

Finally we specify the wakefields. The wakefield functions are read in as tables from input files, that have been generated with specialized programs (e.g. with MAFIA [7] or [8]):

```
SET_SR_WF, file    = '../infiles/srwf_slc.dat'
```

Everything is now set up to track the beam through the lattice:

```
TRACK
```

The end-of-tracking analysis prints a lot of useful data to standard output (e.g. emittance growth, RMS trajectory offsets). Compare the example in section 4.4! In order to study details along the linac, we generate an ASCII table that contains the BPM readings, the emittances, the beta mismatch and other observables as a function of longitudinal position.

```
MEAS_BPM,  file = 'bpm.dat'
```

An external plotting program (e.g. gnuplot or PAW) can now read the ASCII table and generate plots. The table format is described in the reference description of the command MEAS_BPM. Alternatively plots can be generated directly with the command PLOT.

4.6.2 Advanced SLC study

Now we show an example of an advanced SLC simulation. The input file can be found in

```
/afs/slac.stanford.edu/public/software/liar/release/examples/cmdslc2
```

This example includes imperfections, BNS damping, trajectory corrections, feedback loops and emittance bump optimizations. The calculations are performed for 100 statistically different cases. This example illustrates the power of LIAR to study imperfections and their correction.

[6]Alternatively initial beta and alpha functions can be specified

```
  RESET, all                                          ! * Reset data
  SET_CONTROL, debug   = 0,                           ! * Send standard output
               outlun  = 46,                          !   to file 'test.txt' on
               outfile = 'test.txt'                   !   unit 46.
!
  READ_TRNS, infile  = '../infiles/slc.trns',         ! * Read transport input
             energy  = 1.19,                          !   deck and define injection
             npiece  = 1                              !   energy.
!
  SET_RF, energy   = 46.,                             ! * Set final beam energy
          scale    = .t.,                             !   and RF phases. Scale
          kloss    = 0.66d14,                         !   lattice accordingly.
          charge   = 3.5d10,                          !
          phase1   = 22.,                             !
          lswitch1 = 800.,                            !
          phase2   = -16.5                            !
!
  CALC_TWISS, betax   = 3.40,                         ! * Calculate Twiss values.
              betay   = 3.08,                         !   Specify initial beta
              alphax  = 0.156,                        !   and alpha functions.
              alphay  = 0.066                         !
!
  SET_INITIAL, energy   = 1.19,                       ! * Specify initial energy,
               espread  = 0.014,                      !   energy spread, normalized
               nemitx   = 3.d-5,                      !   emittances. Match initial
               nemity   = 0.35d-5,                    !   beta and alpha to Twiss.
               match
!
  SET_BEAM, current  = 3.5d10,                        ! * Set up beam. Specify
            blength  = 1100.d-6,                      !   bunch population, bunch
            nb       = 1,                             !   length, number of
            ns       = 20,                            !   bunches/slices/macro-
            nm       = 3                              !   particles.
!
  SET_SR_WF, file    = '../infiles/srwf_slc.dat'      ! * Define short-range WF's
!
  DEFINE_FEEDBACK, marker = 'feedbk06'                ! * Define feedbacks at
  DEFINE_FEEDBACK, marker = 'feedbk11'                !   predefined markers.
  DEFINE_FEEDBACK, marker = 'feedbk18'                !
!
  TRACK                                               ! * Track beam through.
!
  EMITC_DEF, name = 'ebump1',                         ! * Define emittance bump
             fdbk = 'feedbk06',                       !   using the feedback
             wire = 'wire0001',                       !   'feedbk06' to minimize
             iter = 8,                                !   the emittance at a
             cors = 'slc',                            !   predefined marker.
             xfile = '../infiles/xmatch.ext_elec',    !   Calibrate bumps using
             yfile = '../infiles/ymatch.ext_elec',    !   1-to-1 correction (SLC
             citer = 1, silent                        !   scheme).
!
  EMITC_DEF, name = 'ebump2',                         ! * Define second emittance
             fdbk = 'feedbk11',                       !   bump.
             wire = 'wire0002',                       !
             iter = 6,                                !
             cors = 'slc',                            !
```

```
            xfile = '../infiles/xmatch.ext_elec',     !
            yfile = '../infiles/ymatch.ext_elec',     !
            citer = 1, silent                         !
!
  LOGBOOK, reset                                      ! * Reset the logbook.
  RESET, reference                                    ! * Reset reference data.
!
  DO seed = 1, 100, 100                               ! * Do 100 different cases.
!
     RESET, corrector                                 ! * Reset corrector fields.
     LOGBOOK, set = 1                                 ! * Specify logbook set 1.
!
     ERROR_GAUSS_STRUC, name    = '*',                ! * Randomly misalign all RF
                        x_sigma = 200.e-6,            !   structures.
                        y_sigma = 200.e-6             !
     ERROR_GAUSS_QUAD, name    = '*',                 ! * Randomly misalign quad's
                       x_sigma = 100.e-6,             !   and associated BPM's.
                       y_sigma = 100.e-6,             !
                       bpm     = .t.                  !
     ERROR_GAUSS_BPM, name    = '*',                  ! * Randomly misalign BPM's
                      x_sigma = 100.e-6,              !   with respect to the
                      y_sigma = 100.e-6,              !   quadrupoles.
                      reset   = .f.                   !
!
     TRACKC,  cors = 'slc',                           ! * Track with 1-to-1 corr.
              xfile = '../infiles/xmatch.ext_elec',   !   (SLC scheme). Put results
              yfile = '../infiles/ymatch.ext_elec',   !   into logbook.
              iter  = 1, dolog                        !
     FDBK_GOLD                                        ! * Update feedbk reference.
     LOGBOOK, set = 2                                 ! * Specify logbook set 2.
!
     EMITC, name = 'ebump1', reset                    ! * Optimize emittance bump.
     TRACKFC, cors = 'slc',                           ! * Track with 1-to-1 corr.
              xfile = '../infiles/xmatch.ext_elec',   !   and feedbacks.
              yfile = '../infiles/ymatch.ext_elec',   !
              iter  = 6, citer = 1, silent            !
!
     EMITC, name = 'ebump2', reset                    ! * Optimize emittance bump
     TRACKFC, cors = 'slc',                           ! * Track with 1-to-1 corr.
              xfile = '../infiles/xmatch.ext_elec',   !   and feedbacks. Put
              yfile = '../infiles/ymatch.ext_elec',   !   results into logbook.
              iter  = 6, citer = 1, silent, dolog     !
!
     MEAS_REF, choice='emit', ref=1, reset=.f.        ! * Add emittance to ref 1.
     MEAS_REF, choice='lum',  ref=2, reset=.f.        ! * Add lum. factor to ref 2.
     MEAS_REF, choice='bmag', ref=3, reset=.f.        ! * Add bmag to ref 3.
     PRINT_REF, ref=1, file='out_emit.txt'            ! * Save AV/RMS of ref 1.
     PRINT_REF, ref=2, file='out_lumf.txt'            ! * Save AV/RMS of ref 2.
     PRINT_REF, ref=3, file='out_bmag.txt'            ! * Save AV/RMS of ref 3.
!
     LOGBOOK, process                                 ! * Process the logbook
     LOGBOOK, print, file = 'out_slcemit.log'         ! * Save the logbook file
!
  END DO                                              ! * End of loop over 100 seed
!
  SET_CONTROL, debug   = 0,                           ! * Change standard output
```

```
                outlun   = 6                                    !   to screen
!
```

This simulation will set up the SLC case, implement feedbacks at pre-defined markers and implement SLC-style emittance bumps, that use the feedbacks to cancel dispersion and wakefield dilutions of the beam emittance. Having done the basic setup, LIAR loops over 100 SLC cases with different imperfections. For each case 1-to-1 steering is performed and two emittance bumps are set to minimize the beam emittance at so-called wire markers. A logbook records the emittance after 1-to-1 steering (set 1) and after applying the emittance bumps (set 2). After each seed the logbook is processed to show the average emittance growth for both cases. We also save the average emittance, luminosity factor and beta mismatch along the linac in references, that are printed into ASCII files. We do not go through the details of the simulation and encourage the reader to consult the reference information for each command.

5 LIAR COMMANDS IN THE SIMULATION CONTEXT

The theoretical concepts that were implemented into LIAR were discussed in section 3. Now we concentrate on the corresponding LIAR commands and describe how an actual simulation is put together. A detailed alphabetical description of all commands is supplied in the next section. Here we try to shortly explain the functionality of all commands in the context of a simulation. This is helpful, for determining what commands must be specified and in what order. The following steps are given in the required order for building a simulation run[7]. This section can provide an useful overview on the functionality that is implemented in LIAR.

5.1 Initialization and setup

Before doing anything else the program variables should be initialized to zero. This is done with the command RESET. General control parameters (debug level, standard output, ...) are setup with the command SET_CONTROL.

5.2 Quitting the program

The program LIAR is ended with the commands STOP, QUIT or EXIT.

5.3 Maximum dimensional definitions

Because LIAR is mainly written in FORTRAN77 the maximum dimensions for the lattice, beam, etc. are defined before the compilation of the program. They are all defined in the include file 'dimensions.inc' and can be adjusted easily. However, the program must be recompiled and linked. The command SHOW_DEFINITIONS allows to determine the maximum dimensions for the actual version of LIAR.

5.4 Defining the lattice and the RF

The second step in setting up a simulation is to define a lattice. In version 1.9 lattice files can be read in as TRANSPORT decks[8] or MAD tape files (compare section 8.1). The corresponding commands are READ_TRNS and READ_MAD. Magnet/element types, locations and strengths are defined within the lattice files. Marker points should be added to the lattice files as needed (e.g. for feedbacks). The lattice can be printed using the command PRINT_LATTICE. Marker points, that have been defined, can be listed with the command SHOW_MARKER. The internal consistency of the lattice can be checked with the command CHECK_LATTICE.

RF phases are defined in the input files. In addition the command SET_RF allows to change the RF phases in any part of the accelerator to a given value. In SET_RF also the final beam energy can be adjusted with an appropriate scaling of the lattice. The RF acceleration can be switched off after some specified point with the command RF_SWITCHOFF. The lattice is rescaled and a coasting beam can be simulated.

After having defined the lattice and the RF, additional commands become available. The command READ_SLC allows to read actual SLC magnet strengths and energy profiles and to adjust the RF accordingly. Accelerator support structures (girders with 1 or 2 support points) are defined with the command DEFINE_SUPPORT. All elements that are sensitive to misalignments are being attached to a support structure.

[7]Note that this is the required order for an initial setup of the simulation. Early setup commands can be used again later to modify beam or lattice parameters.

[8]Note that all specified numbers in the TRANSPORT deck are assumed to be real numbers. Integer numbers are not allowed.

This allows to simulate the effect of ground motion, girder vibrations, ... The command SHOW_SUPPORT prints a list of the defined support.

The command SCALE_LATTICE allows to scale all quadrupole and corrector strengths by a specified amount. This is equivalent to change the beam energy and is used in SLC to measure dispersion. Single correctors can be changed in kick angle using the command SET_CORRECTOR.

5.5 Twiss calculation

As soon as the lattice is defined the Twiss parameters can be calculated. The command CALC_TWISS calculates the Twiss parameters for electrons, the command CALC_TWISSP for positrons. The results can be printed to the standard output with the commands PRINT_TWISS and PRINT_TWISSP. It is advisable ALWAYS to calculate the Twiss parameters. They are needed for all kinds of correction schemes, feedbacks, emittance bumps, et cetera.

5.6 Beam setup

Before the actual bunch train is defined the initial beam conditions need to be specified with the command SET_INITIAL. Afterwards the beam is setup with SET_BEAM. When the initial conditions are changed one must not forget to call SET_BEAM again. After the first calls of SET_INITIAL and SET_BEAM only the parameters that change must be respecified. By default the commands use the parameters that were specified for the previous call. SET_BEAM also allows to put random errors on bunch length or charge. If more than one bunch is defined then single bunch charges can be modified separately with the command SET_CHARGE. The BNS autophasing energy spread for the given lattice and beam can be calculated with the command AUTOPHASE.

5.7 Wakefield setup

The wakefields associated with the beam and the accelerating RF structures are read in from separate input files (compare section 8.2). The standard command for reading in transverse and longitudinal short-range wakefields is SET_SR_WF. In a modified version SET_SRQ_WF both dipole and quadrupole short-range wakefields can be specified. Long-range transverse wakefields are specified with the command SET_WF_TRANSV_LR. If wakefields shall be included in the calculation then they must be specified before the relevant tracking command. If the bunch length is changed with SET_BEAM then the wakefields must be redefined! In addition we provide the possibility to specify simplified long-range wakefields via the command SET_LRWF_SA. Internal structure misalignments, however, can not be simulated with those simplified wakefields.

5.8 Accelerator imperfections

Imperfections can be assigned to most beamline parameters in LIAR. The commands ERROR_GAUSS_-STRUC, ERROR_GAUSS_QUAD and ERROR_GAUSS_BPM assign Gaussian errors to the RF structures, the quadrupoles and the BPM's. These commands also allow to offset parameters (e.g. simulating phase ramp). Almost any subset of elements can be selected and BPM's can be misaligned with the quadrupoles. The command MISALIGN_SUPPORT allows to randomly misalign the accelerator support, whereas ATL-MOVE moves the support points assuming the ATL approximation. The command FDBK_MISAL misaligns feedback reference points to the average misalignment of the neighboring BPM's. When feedbacks are used, this command must be called after every change in the BPM alignment. Tables of accelerator imperfections can be printed with the command SHOW_MISALIGN. The command MDLERR allows to implement RF errors that result from imperfections of the drive line for the RF triggers (see command).

5.9 Trajectory feedbacks

Feedbacks (compare section 3.11.6) can be defined at every marker point (compare section 8.1) using the command DEFINE_FEEDBACK. The feedback setpoints are set and reset with the command SET_FEEDBACK. This command also allows to change the active plane of a feedback ('X', 'Y', 'BOTH', 'NONE') giving the possibility to switch off feedbacks. A list of feedbacks is printed from the command SHOW_FEEDBACK. The command FDBK_MISAL references the feedbacks to the BPM misalignment (as in an actual feedback where the BPM readings are fitted). The feedbacks can be golded using the command FDBK_GOLD, which means that the feedback reference information is updated to the actual trajectory offsets at the feedback marker point.

5.10 Multi-device knobs

Multi-device knobs (e.g. N corrector closed bumps) can be defined with the command DEFINE_MULTIKNOB. Devices that can be specified within a multi-device knob include quadrupoles, RF structures and dipole correctors. Parameters that can be changed include transverse misalignments, magnetic fields, longitudinal rotation angles, RF phases and gradients. The value of a multi-device knob is changed with the command SET_MULTIKNOB. Existing multi-device knobs can be displayed using the command SHOW_MULTIKNOB.

5.11 Tracking

When the relevant pieces of information are set up, the command TRACK will track the beam through the beamline. During tracking the bunch positions, the emittances, the BMAG's and the luminosity reductions are calculated and saved for all regular BPM's. The RF BPM's at the beginning and the end of each structure are updated with the beam position. Tracking can be done in first or second order, coupled or uncoupled, with or without the different types of wakefields by specifying the different command parameters. It is recommended always to call the simple TRACK command once after setting up the simulation. The specified options will set up defaults that are used by some correction commands (e.g. by the tracking part of QALIGN). After tracking is done, the command TRAJFIT can be used to fit the beam offset and angle at a specified BPM, using a number of downstream BPM readings.

5.12 Correction algorithms

Two families of correction algorithms are available. There are procedures that do not include tracking. CORRECT performs a 1-to-1 correction by solving a least-squares problem for the BPM's of the whole linac (compare section 3.11.1). The dipole correctors are adjusted accordingly. One needs to call the tracking routine afterwards. CORRECT assumes that exactly one downstream BPM (next QF) is associated to every dipole corrector (this QF). The command might fail if the lattice is not set up properly. Positively charged particles are handled by the command CORRECTP. RFALIGN realigns the RF structures using movers at the two support points of the common girder (compare section 3.11.2). Tracking needs to be called before and after this command.

Another family of commands includes the tracking part. TRACKC will track the beam through the lattice and do a 1-to-1 correction on the fly (compare section 3.11.1). This is a fast and easily converging correction algorithm. The 1-to-1 correction assumes that exactly one downstream BPM (next QF) is associated to every dipole corrector (this QF). The command might fail if the lattice is not set up properly. TRACKCP handles positively charged bunches. TRACKF tracks through the lattice while adjusting the feedbacks at the same time. Finally TRACKFC tracks while adjusting the feedbacks and performing a 1-to-1 correction after feedbacks with zero setpoints. Those commands offer the same options as the TRACK command (compare section 5.11).

QALIGN is another correction algorithm that includes the tracking. It is the most complex optimization routine in LIAR. Based on the BPM readings QALIGN solves a least-squares problem for a beam-based alignment of quadrupoles and RF structures (compare section 3.11.2). This routine involves multiple tracking and the solution of large matrices. It is therefore relatively slow. Beware that QALIGN assumes that there is a BPM at every quadrupole location. This requirement is different than the requirement for 1-to-1 correction. Quadrupoles are assumed to be splitted with the BPM in between. At the border between neighboring sections dipole correctors are needed in order to kick the beam from one section into the next section.

Emittance bumps are another kind of correction algorithm (compare section 3.11.7). The command EMITC_DEF defines an emittance bump. The name of the feedback to be adjusted and the name of a marker where we assume the emittance measurement must be specified. This command will perform some tracking in order to calibrate the specified emittance bump. Alternatively up to two multi-device knobs can be specified with the command EMITC_DEF_MKB in order to optimize the beam emittance. Once the emittance bump is defined, the command EMITC will adjust it to minimize the projected emittance at the observation point. In case multi-device knobs are used, the command EMITC_MKB is available.

It is a good idea always to call the simple TRACK command before the correction commands are used. The specified TRACK options will set up defaults that are used by some correction commands.

5.13 Logbook

An internal logbook keeps tracking results in memory. The DOLOG flag for tracking commands decides whether results are stored in the logbook or not. When looping over simulations we can fill the measurements into a single data set or organize it into several data sets (e.g. a data set for each current with each 100 random seeds). The logbook is managed with the LOGBOOK command. It allows to analyze the data table (mean, sigma) or to print it into an ASCII file.

5.14 Output

The standard ONLINE output of commands can be redirected into any file for later consideration. It contains rather complete informations about parameters, emittance growth, trajectories, et cetera. Summary results are saved and accessed via the logbook. In addition the commands MEAS_BPM and MEAS_EMIT allow to save BPM readings, emittances and a lot of other useful data as a function of s into ASCII files. The command SHOW_MISALIGN saves alignment and related data into ASCII files. The command MEAS_PHASE allows to “measure” the vertical phase advance from a betatron oscillation. It should only be called when a betatron oscillation actually is present in the BPM readings or in any BPM reference (compare section 5.15).

The phase space information of the beam is saved at all marker points. The command MEAS_BUNCH prints out the phase space information of all beam slices for the bunch that was specified with the parameter MBUNCH during the last tracking. The command MEAS_TRAIN provides the multibunch phase space at a specified marker for the whole bunch train.

5.15 Reference data

LIAR allows to save internally reference data for the trajectory, emittance, and similar quantities. At every BPM up to 10 reference informations can be saved. A number of different observables can be saved into specified references with MEAS_REF. For each reference number and each BPM the sum of values, the sum of squared values and the number entries is stored. This allows for example to easily study the average emittance growth at all BPM's. The references can be manipulated with the commands REF_AVG (averaging of a number of references) and REF_SUB (difference between two references). The command PRINT_REF saves a selected reference into a file. Actual SLC data can be read into references using the commands READ_SLC and READ_SLC_ORBIT.

Sometimes it is useful just to specify a few BPM values in a reference trajectory, for example in order to specify a closed bump. This can be done with the command SET_BPMREF. Afterwards TRACKC can be used to steer to the manipulated reference, thus for example implementing the specified closed bump.

5.16 "Experimental" observables

The multi-particle beam dispersion can be determined with the command MEAS_DISPERSION. Optionally files are written that can be used as input to a dispersion-free steering command DFS.

The command DLUM_WAIST estimates the expected waist shift due to the simulated linac errors and its effect on the luminosity.

The linac optical functions (phase advance, beta functions, alpha functions, amplitude growth) can be determined beyond the single-particle Twiss values with the command MEAS_PHASE2. They are obtained including wakefield effects (multi-particle beam dynamics) and lattice errors.

The sensitivity of the final beam offset and angle to beam deflections or quadrupole offsets is obtained with the commands SENS_YKICK or SENS_YQUAD.

5.17 Executing command files

Command files (lists of valid LIAR commands) can be loaded and executed with the LOAD, EXEC or READ commands. Command files allow the automatic execution of nested command loops. They may not be called from within another command file.

5.18 Additional output files

Some commands allow to write a single line result into an output file that is specified by an output unit number. This for example allows to change a beam or measurement parameter in steps and to record an important observable as a function of this parameter. The corresponding output files are opened and closed with the commands OPEN_FILE and CLOSE_FILE.

5.19 Graphics

LIAR provides an interface to the *gnuplot* [15] scientific plotting program. The use of the external *gnuplot* program is completely transparent for the user and does not require any knowledge of this program. LIAR provides ready-to-plot input data and command files. It automatically calls the graphics program, controls the graphics output (screen, PS, EPS, PS-printer) and will return to the main LIAR program. In order to allow for advanced font and border handling we assume that the beta release version 3.6 (or later) of gnuplot is installed. This beta release is freely available for almost any computer platform (including AIX 3.2, 32-bit windows, LINUX, etc.). A LIAR PLOT command allows to create plots of almost any internal information.

Axis and key labels including units are automatically created. Up to two curves can be overlayed in a single plot. More details of available options are described in the reference section under PLOT.

Alternatively, the LIAR interface to MATHEMATICA (compare section 4.5) can be used for plotting purposes.

6 COMMAND REFERENCE SECTION

Here we give an alphabetical list of commands. For each command the available options and their default values are explained. Several commands need to be executed in the right order. For example, we have to define the lattice and initial beam conditions before we can track the beam. Other interdependencies might be less obvious but can be figured out with the information from the previous section.

ATLMOVE

Move the accelerator support assuming the ATL approximation [9]:

$$\sigma^2 = A \cdot T \cdot L \tag{101}$$

where T is the time in seconds and L is the distance in meters.

AX	0.	Constant A [$\mu m^2/m/s$ for the horizontal plane.]
AY	0.	Constant A [$\mu m^2/m/s$ for the vertical plane.]
T	0.	Time in seconds.
SEED	Time	Seed for random number generator.
CUT	3.	Cutoff in number of sigmas for ATL drifts.
RESET	.F.	Logical flag for resetting previous misalignments.

AUTOPHASE

Estimate the BNS autophasing energy spread σ_E/E for a simple two-particle model:

$$\frac{\sigma_E}{E} = A \cdot I \cdot \sigma_z \cdot L^2 \cdot \frac{\csc{(\phi/2)}^2}{4E} \tag{102}$$

Here, E is the beam energy in GeV, σ_z is the RMS bunch length in m, I is the bunch population in 10^{10}, ϕ is the phase advance per FODO cell, L is the quadrupole spacing in m and A is the wakefield slope in Gev/m^3/(10^{10} ppb). The results are written to a file.

FILE	'autphase.bns'	Name for output file.
WFA	145.	Wakefield slope [Gev/m^3/(10^{10} ppb)].

The output file provides the following information:

```
-------------------------------------------------------
  Column  |  Information
    Nb.   |
-------------------------------------------------------
     1    | Quadrupole number
     2    | S-position [m]
     3    | Absolute energy spread [GeV] (result)
     4    | Relative energy spread (result)
     5    | Phase advance since previous quadrupole
     6    | Distance to previous quadrupole
-------------------------------------------------------
```

CALC_TWISS

Calculate e- Twiss values for the actual lattice and optics. The Twiss values are determined from the transfer matrices R (zero bunch length). The RF acceleration can be roughly corrected for single-bunch beam loading effects. Assume electrons. The command can take its initial values from the Twiss array (e.g. only if specified in MAD input deck).

KLOSS	0.	Beam loading loss factor in V/C/m.
CHARGE	0.	Number of particles per bunch for beam loading.
PSIX	Twiss	Initial x phase advance in rad.
PSIY	Twiss	Initial y phase advance in rad.
BETAX	Twiss	Initial β_x in m.
BETAY	Twiss	Initial β_y in m.
ALPHAX	Twiss	Initial α_x.
ALPHAY	Twiss	Initial α_y.
ERRORS	.f.	Flag for inclusion of errors.

CALC_TWISSP

Calculate the e+ Twiss values for the actual lattice and optics. Same algorithm as for electrons (compare CALC_TWISS).

KLOSS	0.	Beam loading loss factor in V/C/m.
CHARGE	0.	Number of particles per bunch for beam loading.
PSIX	Twiss	Initial x phase advance in rad.
PSIY	Twiss	Initial y phase advance in rad.
BETAX	Twiss	Initial β_x in m.
BETAY	Twiss	Initial β_y in m.
ALPHAX	Twiss	Initial α_x.
ALPHAY	Twiss	Initial α_y.
ERRORS	.f.	Flag for inclusion of errors.

CHECK_LATTICE

Check the lattice for errors/inconsistencies.

CORRECT

Standard 1-to-1 trajectory correction where a least-squares solution for the whole linac is implemented. The dipole correctors are set to their new values. This correction depends on the zero current Twiss values. Therefore it is recommended that a first correction is done with the command TRACKC.

PLANE	'both'	Plane for correction ('X', 'Y', 'BOTH').
BUNCH	0	Selection of bunch for correction (0 = all).
CORS	'standard'	Selection of corrector scheme ('STANDARD', 'SLC').
XFILE	-	X-Corrector, BPM association file for option 'SLC'.
YFILE	-	Y-Corrector, BPM association file for option 'SLC'.
BPMX	'qf'	Selection of BPM's for horizontal plane ('QF','QD','ALL','*').
BPMY	'qd'	Selection of BPM's for vertical plane ('QF','QD','ALL','*').
SCALE	1.	Scale factor for correction to be applied.
REJECT	1.	Threshold for kick rejection in rad.

The standard correction scheme uses correctors at QF's for the horizontal correction and correctors at QD's for the vertical correction. Correction schemes may fail due to specifics in the used lattice. A new lattice generally has to be made working with this routine! BPM's can be specified with a search string, e.g. BPMY = 'MY*' will select all BPM's with names that start with MY. Note that the CORS option 'SLC' reads in an input file with SLC definitions.

CORRECTP

Standard 1-to-1 trajectory correction (like CORRECT) for positrons.

PLANE	'both'	Plane for correction ('X', 'Y', 'BOTH').
BUNCH	0	Selection of bunch for correction (0=all).
CORS	'standard'	Selection of corrector scheme ('STANDARD', 'SLC').
XFILE	-	X-Corrector, BPM association file for option 'SLC'.
YFILE	-	Y-Corrector, BPM association file for option 'SLC'.
BPMX	'qf'	Selection of BPM's for horizontal plane ('QF','QD','*').
BPMY	'qd'	Selection of BPM's for vertical plane ('QF','QD','*').
SCALE	1.	Scale factor for correction to be applied.
REJECT	1.	Threshold for kick rejection in rad.

CLOSE_FILE

Close an ASCII file that previously was opened with the command OPEN_FILE.

UNIT	0	Unit for file number to close.

DEFINE_FEEDBACK

A feedback loop is defined. The command evaluates the effectiveness of all pairs of two correctors upstream of the observation point, where one of the two correctors is the previous or the pre-previous corrector. The most effective pair is assigned to the feedback loop.

MARKER	-	Name of the observation marker for the feedback (must be set).
NCOR	8	Number of upstream dipole correctors to consider.
PLANE	'both'	Active plane for the feedback loop.
BUNCH	1	Active bunch for the feedback loop (0 = all).

DEFINE_MULTIKNOB

A multi-knob is defined. A multi-knob has a number of elements attached to it. Each element has a defined parameter that is to be moved by a specified scale factor. A simple case of a multi-knob would be a closed three corrector bump. In this case three dipole correctors would be assigned to the multi-knob. The relevant parameters are the corrector fields. Each corrector has a scale factor that is calculated such that the total bump is closed. The command SET_MULTIKNOB assigns a value to the multiknob. Note that multi-knobs can be defined with quadrupoles, structures and correctors. The parameters can be field, offset, tilt, RF-phase or energy gradient.

NAME	' '	Name of the multi-knob (must be set).
FILE	' '	Optional input file for multi-knob definitions (not yet implemented).
CORRECT	.F.	Switch for 1-to-1 correction in multi-knob region.
ELEMENT	' '	Name of attached element (must be set).
PARAMETER	' '	Parameter of element to be varied (must be set).
FACTOR	0.	Scale factor for parameter (must be set).

Note that the ELEMENT/PARAMETER/FACTOR block must be repeated multiple times in order to assign several elements to a multi-knob. However, it must be repeated as a block. Here we list the elements and parameters that are recognized:

```
------------------------------------------------------------
  Element type    |  Recognized parameters
------------------------------------------------------------
  Quadrupoles     |  X            ->  X offset
                  |  Y            ->  Y offset
                  |  TILT         ->  Tilt angle
                  |  FIELD        ->  Quadr. field
------------------------------------------------------------
  Structures      |  X            ->  X offset
                  |  Y            ->  Y offset
                  |  PHASE        ->  RF phase
                  |  GRADIENT     ->  Acclerating gradient
------------------------------------------------------------
  Dipole corr-    |  FIELD        ->  Dipole field
     ectors       |  TILT         ->  Tilt angle
------------------------------------------------------------
```

DEFINE_SUPPORT

Define the support for the accelerator. Assuming a regular lattice any number of structures can be assigned to a single girder. Quadrupoles can be assigned to the same girders or can be defined with a separate support. Every girder with structures on it has two support points (begin and end). A quadrupole (plus BPM, corrector, ...) has only one support point. For example, this allows the easy simulation of ground motion.

NGIRDER	0	Number of RF structures per girder (must be set).
QSUPPORT	.T.	Logical flag for a separate quadrupole support.

DFS

Routine for calculation of dispersion-free steering (DFS) solution from an absolute orbit and one or more dispersion orbits. The files are expected to have been generated with the command MEAS_DISPERSION. The DFS is done over the range of BPM's and correctors in the input files. Note that the reference is only updated for the BPM's used for DFS (at QF's or QD's) and only in the provided range.

TRAJFILE	' '	Input file for the absolute trajectory.
DFSFILE1	' '	Input file nb. 1 for the dispersion.
DFSFILE2	' '	Input file nb. 2 for the dispersion.
DFSFILE3	' '	Input file nb. 3 for the dispersion.
DFSFILE4	' '	Input file nb. 4 for the dispersion.
DFSFILE5	' '	Input file nb. 5 for the dispersion.
DFSFILE6	' '	Input file nb. 6 for the dispersion.
DFSFILE7	' '	Input file nb. 7 for the dispersion.
DFSFILE8	' '	Input file nb. 8 for the dispersion.
DFSFILE9	' '	Input file nb. 9 for the dispersion.
DOCOR	.t.	Flag to decide whether to set the correctors.
REF	1	Reference number to be updated with the DFS trajectory.

DLUM_WAIST

Calculate the luminosity loss from IP waist shifts due to the linac BMAG. Calculated are the luminosity loss (RMS and maximum) as an average over single slices and for the whole bunch (assuming that slice to slice offsets have not been corrected at the IP).

FILE	'dlum_waist.out'	Output file.
BUNCH	1	Bunch to be considered (0=all).
UNIT	0	If UNIT is not equal 0 one line with the results is written to the file unit 'UNIT'. The file must be opened with OPEN_FILE and closed with CLOSE_FILE.

EMITC

Calculate the optimal settings of the specified emittance bump in order to minimize emittance. This is a deterministic calculation. Details are explained in section 3.11.7.

NAME	-	Name of emittance bump to use (must be set).
SETFDBK	.T.	Flag whether to set the feedbacks accordingly.
RESET	.F.	Flag for resetting of old feedback setpoints.

EMITC_MKB

As EMITC but allows to optimize the emittance with feedbacks or multi-knobs.

NAME	-	Name of emittance bump to use (must be set).
SETFDBK	.T.	Flag whether to set the feedbacks accordingly.
RESET	.F.	Flag for resetting of old feedback setpoints.

EMITC_DEF

Define an emittance bump and prepare emittance optimization. Details are explained in section 3.11.7. For the calibration of the emittance bump the beam must be tracked through the lattice. After feedbacks with zero setpoints a 1-to-1 correction is performed. Compare the command EMITC.

NAME	-	Name for the emittance bump (must be set).
MBUNCH	1	Active bunch for emittance optimization.
FDBK	-	Name of feedback loop to be used (must be set).
WIRE	-	Name of MARKER for “wire measurement” (must be set).
ITER	3	Number of iterations for bump calibration.
WF_T_SR	.T.	Flag for transverse short-range wakefields.
WF_T_LR	.T.	Flag for transverse long-range wakefields.
WF_L_SR	.T.	Flag for longitudinal short-range wakefields.
WF_L_LR	.T.	Flag for longitudinal long-range wakefields.
RFBUNCH	1	Bunch selection for RF BPM.
CITER	1	Number of 1-to-1 correction iterations.
CORS	'standard'	Selection of corrector scheme ('STANDARD', 'SLC').
XFILE	-	X-Corrector, BPM association file for option 'SLC'.
YFILE	-	Y-Corrector, BPM association file for option 'SLC'.
BPMX	'qf'	Selection of BPM's for horizontal plane ('QF','QD','*').
BPMY	'qd'	Selection of BPM's for vertical plane ('QF','QD','*').
MINPHASE	$\pi/4$	Minimum phase advance between corrector and associated BPM.
MAXPHASE	$3\pi/4$	Maximum phase advance between corrector and associated BPM.
DPOS	500.d-6	Offset in m for emittance bump calibration.
DANG	50.d-6	Angle in radian for emittance bump calibration.
SILENT	.F.	Logical flag for silent mode (reduced standard output).

EMITC_DEF_MKB

As EMITC_DEF but the emittance bump can be defined with multi-knobs as well. Note that EMITC_DEF_MKB does not perform any 1-to-1 correction after feedbacks or multi-knobs. E.g. non-closure of bumps etc. is NOT corrected. Compare the command EMITC_MKB. Note that FDBK or MBX1_NAME, MBX2_NAME, MBY1_NAME and MBY2_NAME must be specified. A feedback has four degrees of freedom: x, x', y, y'. Every degree of freedom is replaced by a multi-knob, thus requiring four multi-knobs (two per plane).

NAME	-	Name for the emittance bump (must be set).
MBUNCH	1	Active bunch for emittance optimization.
FDBK	-	Name of feedback loop to be used (must be set if not multi-knob).
MBX1_NAME	-	Name of 1st X multi-knob (must be set if not feedback).
MBX2_NAME	-	Name of 2nd X multi-knob (must be set if not feedback).
MBY1_NAME	-	Name of 1st Y multi-knob (must be set if not feedback).
MBY2_NAME	-	Name of 2nd Y multi-knob (must be set if not feedback).
WIRE	-	Name of MARKER for "wire measurement" (must be set).
ITER	3	Number of iterations for bump calibration.
WF_T_SR	.T.	Flag for transverse short-range wakefields.
WF_T_LR	.T.	Flag for transverse long-range wakefields.
WF_L_SR	.T.	Flag for longitudinal short-range wakefields.
WF_L_LR	.T.	Flag for longitudinal long-range wakefields.
RFBUNCH	1	Bunch selection for RF BPM.
CITER	1	Number of 1-to-1 correction iterations.
CORS	'standard'	Selection of corrector scheme ('STANDARD', 'SLC').
XFILE	-	X-Corrector, BPM association file for option 'SLC'.
YFILE	-	Y-Corrector, BPM association file for option 'SLC'.
BPMX	'qf'	Selection of BPM's for horizontal plane ('QF','QD','*').
BPMY	'qd'	Selection of BPM's for vertical plane ('QF','QD','*').
MINPHASE	$\pi/4$	Minimum phase advance between corrector and associated BPM.
MAXPHASE	$3\pi/4$	Maximum phase advance between corrector and associated BPM.
DPOS	500.d-6	Offset in m for emittance bump calibration.
DANG	50.d-6	Angle in radian for emittance bump calibration.
SILENT	.F.	Logical flag for silent mode (reduced standard output).

ERROR_GAUSS_BPM

Random Gaussian misalignment of the BPM's.

NAME	'*'	Name of BPM's to misalign. '*' specifies all. The string can be used for searching on the names (e.g. 'M02*').
SEED	Time	SEED for random misalignments.
RESET	.T.	Flag for resetting previously assigned BPM errors to zero.
RESOL	0.	BPM resolution in m.
X_MEAN	0.	Average horizontal misalignment in m.
X_SIGMA	0.	RMS horizontal misalignment in m.
X_CUT	3.	Sigma cut on Gaussian tails.
Y_SIGMA, ...	0.	RMS vertical misalignment in m.

ERROR_GAUSS_QUAD

Assign random Gaussian imperfections to the quadrupoles.

NAME	'*'	Name of quadrupoles to misalign. '*' specifies all. The string can be used for searching on the names (e.g. 'QF*').
SEED	Time	SEED for random misalignments.
RESET	.T.	Flag for resetting previously assigned quad errors to zero.
BPM	.T.	Flag for misaligning BPM's within split quad's at the same time.
X_MEAN	0.	Average horizontal misalignment in m.
X_SIGMA	0.	RMS horizontal misalignment in m.
X_CUT	3.	Sigma cut on Gaussian tails.
Y_SIGMA, ...	0.	RMS vertical misalignment in m.
T_SIGMA, ...	0.	RMS rotation error around the longitudinal axis in rad.
DK_SIGMA, ...	0.	RMS strength error $\Delta K/K$.

ERROR_GAUSS_STRUC

Assign random Gaussian imperfections to the RF structures.

NAME	'*'	Name of structures to misalign. '*' specifies all. The string can be used for searching on the names (e.g. 'K02*').
SEED	Time	SEED for random misalignments.
RESET	.T.	Flag for resetting previously assigned structure errors to zero.
SPLIT	.F.	Flag for optional splitting of combined structures (SLC).
RESOL	0.	Resolution of RF BPM's.
NUM_ERR_TYPE	1	Number of error types for structures.
X_MEAN	0.	Average horizontal misalignment of whole structure in m.
X_SIGMA	0.	RMS horizontal misalignment of whole structure in m.
X_CUT	3.	Sigma cut on Gaussian tails.
Y_SIGMA ...	0.	RMS vertical misalignment of whole structure in m.
PX_SIGMA ...	0.	RMS horizontal misalignment of pieces in m.
PY_SIGMA ...	0.	RMS vertical misalignment of pieces in m.
PHAS_SIGMA ...	0.	RMS phase error in radian.
GRAD_SIGMA ...	0.	RMS gradient error $\Delta G/G$.

EXEC

Execute a command file. This command is identical to 'READ' and 'LOAD'. It can only be used from the interactive command level. Do not use it within a command file! LIAR will read all commands from the specified file and switch back to the interactive program level at its end.

FILE - Name of input command file (must be specified).

EXIT

Exit the program. This command is identical to 'STOP' and 'QUIT'.

FDBK_GOLD

Gold the feedbacks. The feedback reference information is updated to the actual trajectory offset and angle in the feedback marker.

BUNCH	0	Bunch number of updating information (0=average).
RESET	.T.	Reset all setpoints to zero.

FDBK_MISAL

The feedbacks are misaligned to the neighbouring BPM's. This command allows the feedbacks to follow the beamline misalignment, therefore only acting on the beam offset with respect to the beamline misalignment (as in a real feedback). Actually the feedback reference information is adjusted.

NBPM	4	Number of up- and downstream BPM's to consider for beamline misalignment.
RESET	.T.	Flag for resetting previously assigned reference information.

GNUPLOT

See the command PLOT!

LOAD

Execute a command file. This command is identical to 'READ' and 'EXEC'. It can only be used from the interactive command level. Do not use it within a command file! LIAR will read all commands from the specified file and switch back to the interactive program level at its end.

FILE	-	Name of input command file (must be specified).

LOGBOOK

The logbook command allows to save and analyze results. It is filled during tracking with emittance growth, luminosity reduction and RMS orbits. At the end of the job the average and RMS over all the stored results can be calculated. Alternatively all data can be written into an output file and analyzed for average and RMS values. Separate sets of data can be defined. Most options of the command should be used alternatively and cannot be combined.

FILE	-	Name for output file if writing to file.
PRINT	.F.	Write data into the file specified by FILE.
RESET	.F.	Reset the logbook.
SET	0	Change the actual set number to the value of SET. Zero has no effect. Different sets are analyzed separately and are appended into a single output file.
NEWSET	.F.	Increase the actual number of the data set by one.
PROCESS	.F.	Process logbook data and write analysis to unit OUTLUN.

The output file provides the following information (note that the different sets are appended):

```
-----------------------------------------------------
  Column   |  Information
  Number   |
-----------------------------------------------------
     1     | Number of seed
     2     | Horizontal emittance growth [%]
     3     | Vertical emittance growth [%]
     4     | Horizontal luminosity reduction
     5     | Vertical luminosity reduction
     6     | Horizontal RMS trajectory offset [um]
     7     | Vertical RMS trajectory offset [um]
     8     | Final beam energy [GeV]
     9     | Horizontal BMAG
    10     | Vertical BMAG
-----------------------------------------------------
```

MDLERR

Implement an RF phase error that goes from "-ERROR" at the beginning of the linac to "+ERROR" at the end. The error changes linearly with longitudinal position s. This is a main driveline (MDL) type of error. Assume that the phase between the RF and the beam is right in the beginning. Errors along the drive line for the RF triggers will increase linearly with the length. This error runs from 0 to "2*ERROR". However, the energy spread is kept at a minimum in the linac, using a global RF phase knob ("phaseramp"). This will put the average RF phase error to zero. We obtain the type of error that is implemented with MDLERR.

ERROR	0.d0	Magnitude of error (degree).
RESET	.F.	Logical flag for resetting of previous errors.

MEAS_BPM

The actual BPM values are read and saved into an output ASCII file. The BPM readings can be normalized with the beta functions and the beam energy.

FILE	-	Name for BPM output file (must be set).
ERROR	.T.	Flag whether to include BPM errors or not.
NORM	.F.	Flag whether to normalize BPM readings with Twiss or not.
BUNCH	1	Bunch number for BPM measurement (0=bunch average weighted with absolute charge).

The output file provides the following information:

```
-------------------------------------------------------
  Column  |  Information
  Number  |
-------------------------------------------------------
     1    | BPM number
     2    | S-position [m]
     3    | Horizontal BPM reading [m]
     4    | Vertical BPM reading [m]
     5    | Horizontal normalized emittance [m-rad]
     6    | Vertical normalized emittance [m-rad]
     7    | Horizontal phase advance [rad]
     8    | Vertical phase advance [rad]
     9    | Horizontal beta function [m]
    10    | Vertical beta function [m]
    11    | Horizontal BMAG
    12    | Vertical BMAG
    13    | Horizontal luminosity factor [%]
    14    | Vertical luminosity factor [%]
    15    | Beam energy [GeV]
    16    | Relative beam energy spread
    17    | Horizontal beam size [m]
    18    | Horizontal divergence [rad]
    19    | Vertical beam size [m]
    20    | Vertical divergence [rad]
    21    | Absolut energy spread [GeV]
-------------------------------------------------------
```

MEAS_BUNCH

At each marker location the phase space information for a single bunch is available. This bunch is specified with the 'MBUNCH' option in the tracking commands. The command 'MEAS_BUNCH' allows to print the phase space information for all beam slices at a marker position into an ASCII file. For printing a list of available markers refer to the command 'SHOW_MARKER'. Markers must be defined in the input deck!

FILE	-	Name for output file (must be set).
MARKER	-	Name of marker (must be set).
BUNCH	1	Bunch number to be used for retrieving internal z-info.

The output file provides the following information:

```
------------------------------------------------------
 Column  |  Information
 Number  |
------------------------------------------------------
    1    | Slice number
    2    | Horizontal slice offset X [m]
    3    | Horizontal slice angle X' [rad]
    4    | Vertical slice offset Y [m]
    5    | Vertical slice angle Y' [rad]
    6    | Horizontal norm. slice emittance [m-rad]
    7    | Vertical norm. slice emittance [m-rad]
    8    | Slice energy [GeV]
    9    | Slice charge [C]
   10    | Longitudinal slice position z [m]
------------------------------------------------------
```

MEAS_DISPERSION

Measure the dispersion by changing the beam energy or by changing the lattice strength. Optionally write file to be used as input to the dispersion-free steering command DFS.

ERROR	0.001	Fractional error on beam energy. This is converted into a lattice strength error.
INIT	.t.	Flag for establishing nominal reference by calling tracking. Alternatively any BPM reference or the actual BPM information can be used as nominal reference.
UNIT	0	If unit is not equal zero the dispersion results are written into the File with unit number 'UNIT'. The file must be opened and closed with the command OPEN_FILE and CLOSE_FILE.
SILENT	.f.	Silent option for command.
CHOICE	'energy'	Choice of method to determine dispersion. Valid choices are 'energy' and 'lattice'.
BUNCH	0	Bunch for dispersion measurement (0=all).
DESIGN	.f.	Measure dispersion by adjusting errors or by changing the nominal values. Gives the same results but can be important for calculation of Twiss.
RESTORE	.t.	Flag for restoring the original state.
FILE	' '	Optional output file for dispersion vs z.
REF	0	If REF is not equal zero then the nominal trajectory is read from the BPM reference 'REF'.
SMIN	0.	Minimum s value in m for dispersion measrement.
SMAX	end	Maximum s value in m for dispersion measurement.
DFS	.f.	Flag for preparing DFS input file. Everything else are DFS options.
CHECK	.f.	Flag for printing out check of BPM-corrector association.
DFSFILE	'dfsfile'	File name for DFS file to be written.
CORS	'standard'	Corrector scheme to be used ('standard', 'SLC').
BPMX	'QF'	Definition of BPM's for x-correction ('QD', 'QF', 'ALL').
BPMY	'QD'	Definition of BPM's for y-correction ('QD', 'QF', 'ALL').
MINPHASE	$\pi/4$	Minimum phase advance between corrector and associated BPM.
MAXPHASE	$3\pi/4$	Maximum phase advance between corrector and associated BPM.
XFILE	-	X-Corrector, BPM association file for option 'SLC'.
YFILE	-	Y-Corrector, BPM association file for option 'SLC'.

The output file specified with the FILE parameter provides the following information:

```
--------------------------------------------------
 Column  |  Information
```

```
  Number   |
-----------------------------------------------------
     1     | BPM number
     2     | S-position [m]
-----------------------------------------------------
```

If data is written to the file specified by UNIT then for each call one line containing the following information is appended.

```
-----------------------------------------------------
  Column   |  Information
  Number   |
-----------------------------------------------------
     1     | Energy deviation for measurement
     2     | Final X dispersion [m]
     3     | Final Y dispersion [m]
     4     | RMS X dispersion along linac [m]
     5     | RMS Y dispersion along linac [m]
-----------------------------------------------------
```

MEAS_EMIT

The actual emittances at the BPM's are read and saved into an ASCII output file.

FILE	-	Name for emittance output file (must be set).
BUNCH	1	Bunch number for emittance measurement (0=all).

The output file provides the following information:

```
-----------------------------------------------------
  Column   |  Information
  Number   |
-----------------------------------------------------
     1     | BPM number
     2     | S-position [m]
     3     | Horizontal normalized emittance [m-rad]
     4     | Vertical normalized emittance [m-rad]
-----------------------------------------------------
```

MEAS_PHASE

The vertical phase advance on a betatron oscillation (must be generated prior to this command by SET_INITIAL or SET_CORRECTOR and TRACK) is measured by detecting the locations of the zero-crossings. The method used is identical to the one used at the SLC lattice diagnostic pulse (LDP). The results are saved into an ASCII file.

FILE	-	Name for phase advance output file (must be set).
SLCFILE	-	Name for phase advance output file in SLC-LDP format.
BUNCH	1	Bunch number for phase advance measurement (0=all).
ERROR	.T.	Flag whether to include BPM reading errors.
REF	0	The phase can be measured from BPM reference “REF”.
PLANE	'Y'	Plane for phase measurement (X,Y).

The output file provides the following information:

```
------------------------------------------------------
  Column  |  Information
    Nb.   |
------------------------------------------------------
     1    | S-position of BPM's [m]
     2    | Measured vertical phase advance [rad]
     3    | Twiss model vertical phase advance [rad]
     4    | Difference between 2 and 3 [rad]
     5    | Vertical beta function [m]
     6    | Vertical traj. offset (incl. errors) [m]
     7    | Beam energy [GeV]
------------------------------------------------------
     8    | S-position of zero crossing [m]
     9    | Meas. phase advance at zero crossing [rad]
    10    | Difference with respect to model [rad]
------------------------------------------------------
```

MEAS_PHASE2

This commands simulates an intial beam offset and angle and determines the linac optical functions from the resulting BPM offsets. The method is described in section 3.15. Both planes are diagnosed at once. The results are saved into an ASCII file. In contrary to the command MEAS_PHASE this command does not require any preceding implementation of a betatron oscillation.

FILE	'phase2.dat'	Name for output file.
BUNCH	1	Bunch number for phase advance measurement (0=all).
SAVETWISS	.F.	Flag for saving Twiss functions at BPM's into file 'bpmtwiss.dat'.
NFIT	11	Number of downstream BPM's for (x, x') fit.
NCUT	3	Number of RMS cut for elimination of bad BPM's.
FUDGE	1.0	Possible wakefield fudge factor for single-particle phase advance.

The output file provides the following information:

```
-------------------------------------------------------
  Column  |  Information
    Nb.   |
-------------------------------------------------------
     1    | Number of BPM
     2    | S-position of BPM [m]
     3    | X linac phase advance [rad]
     4    | X diff. phase adv. from linac to Twiss [rad]
     5    ! X amplification (1.0 without WF's)
     6    | X beta mismatch (BMAG)
     7    | Linac X beta function [m]
     8    | Design X beta function [m]
     9    | Diff. in X alpha function
    10    | Y linac phase advance [rad]
    11    | Y diff. phase adv. from linac to Twiss [rad]
    12    ! Y amplification (1.0 without WF's)
    13    | Y beta mismatch (BMAG)
    14    | Linac Y beta function [m]
    15    | Design Y beta function [m]
    16    | Diff. in Y alpha function
-------------------------------------------------------
```

MEAS_REF

At each BPM a number of different references (e.g. number 1 to 10) can be stored internally. This allows for example to calculate the average X and Y emittance growth along the linac for a number of seeds. The command MEAS_REF takes a specified observable (X and Y) and stores it into the selected reference. By default each reference stores the sum, the sum of the squares and the number of entries. The option RESET allows to reset a reference to zero before it is filled. The option SUBREF allows to subtract another reference before saving the data.

CHOICE	BPM	Choice for observable to be put into reference. Possible choices are: BPM - beam offsets (X and Y) at the BPM's. EMIT - emittances (X and Y) at the BPM's. LUM - luminosity factors (X and Y) at the BPM's. ENERGY - beam energy at the BPM's. ESPREAD - beam energy spread at the BPM's. BMAG - beta mismatch (X and Y) at the BPM's. BSIZE - beam sizes (X and Y) at the BPM's. DIVERGENCE - beam divergences at the BPM's.
ERROR	.T.	Logical flag for beam offsets to include BPM offset errors.
RESET	.F.	Logical flag for resetting the reference to zero before filling it.
SILENT	.F.	Logical flag for silent mode (no standard output).
BUNCH	0	Choice of bunch for observable (0 = all).
REF	1	Number of reference to fill.
SUBREF	0	Number of reference to subtract before filling reference number REF (0 = none).

MEAS_TRAIN

At each marker location the bunch phase space information is available for the whole bunch train. This command prints the phase space information for all bunches at a marker position into an ASCII file. For a list of available markers refer to the command 'SHOW_MARKER'.

FILE	-	Name for output file (must be set).
MARKER	-	Name of marker (must be set).

The output file provides the following information:

```
------------------------------------------------------
 Column  |  Information
   Nb.   |
------------------------------------------------------
     1   | Bunch number
     2   | Horizontal trajectory offset X [m]
     3   | Horizontal trajectory angle X' [rad]
     4   | Vertical trajectory offset Y [m]
     5   | Vertical trajectory angle Y' [rad]
     6   | Horizontal norm. bunch emittance [m-rad]
     7   | Vertical norm. bunch emittance [m-rad]
     8   | Bunch energy [GeV]
     9   | Bunch charge [C]
    10   | Horizontal whole beam emittance [m-rad]
    11   | Vertical whole beam emittance [m-rad]
------------------------------------------------------
```

MISALIGN_SUPPORT

Randomly misalign the accelerator support that was previously defined with the command DEFINE_SUPPORT.

SIGX	0.	Horizontal RMS misalignment [m].
SIGY	0.	Vertical RMS misalignment [m].
SEED	Time	Seed for random number generator.
CUT	3.	Cutoff in number of sigmas for misalignments.
RESET	.F.	Reset old misalignments.
QUAD	.T.	Misalign quadrupole support points.
STRUC	.T.	Misalign structure support points.

OPEN_FILE

Open an ASCII file.

UNIT	0	Unit for file number to open.
FILE	' '	File name.

PLOT

The PLOT command provides for scientific and ready-to-publish quality plots through the *gnuplot* program. The use of the external *gnuplot* program is completely transparent for the user and does not require any knowledge of this program. Gnuplot [15] is freely available for almost any computer platform and LIAR assumes that the gnuplot 3.6 beta release is installed in the path of the program. The LIAR graphics allows to graphically visualize almost any internal information in plots. The output is send to the screen or/and to PS or EPS files or/and to a postscript printer.

1. Horizontal axis:

SELECT	'BPM'	Selection of elements to be used for plots. This parameter puts constraints on valid choices for the horizontal and vertical axes. Valid selections are listed later.
X	'S'	Choice for horizontal axis. Valid selections are listed later.
XPLANE	'X'	Selected plane for horizontal axis.
XMIN	automatic	Minimum value for X-axis (automatic if not specified).
XMAX	automatic	Maximum value for X-axis (automatic if not specified).
MARKER	'LAST'	Name of marker for SELECT = 'MARKER'.

2. First vertical axis:

Y1	'TRAJ'	Choice for first vertical axis (left). Valid selections are listed later.
PLANE1	'X'	Selected plane for first vertical axis.
BUNCH1	1	Bunch selection for first vertical axis.
Y1MIN	automatic	Minimum value for first Y-axis (automatic if not specified).
Y1MAX	automatic	Maximum value for first Y-axis (automatic if not specified).
TITLE1	automatic	Title for first Y data selection (automatic if not specified).
LINTYPE1	'lines'	Choose plot type for first data series. Choose from: lines, points, linespoints, impulses, dots, steps, fsteps, histeps, boxes, boxerrorbars, financebars, candlesticks or vector. This is passed directly to gnuplot and therefore it is important to type in lower letters!

3. Second vertical axis:

Y2	' '	Choice for second vertical axis (right). If not specified then there is only one set of data. Valid selections are listed later.
PLANE2	'Y'	Selected plane for second vertical axis.
BUNCH2	1	Bunch selection for second vertical axis.
Y2MIN	automatic	Minimum value for second Y-axis (automatic if not specified).
Y2MAX	automatic	Maximum value for second Y-axis (automatic if not specified).
TITLE2	automatic	Title for second Y data selection (automatic if not specified).
LINTYPE2	'lines'	Choose plot type for second data series. Compare to LINTYPE1!

4. Output options:

TERM	.T. or .F.	Display graphics online on the terminal. Default is true for LIAR in interactive mode and false for the batch mode.
PS	.F. or .T.	Save graphics into a Postscript (PS) file. Default is false for LIAR in interactive mode and true for the batch mode.
PSFILE	'LIAR.ps'	Name of Postscript file.
EPS	.F.	Save graphics into Encapsulated Postscript (EPS) file.
EPSFILE	'LIAR.eps'	Name of Encapsulated Postscript file.
PRINTER	-	Name of printer to send the Postscript file to (only UNIX).

5. Plot options:

STYLE	automatic	Two plot styles are available: 1) 'ONE' - All data is plotted versus a single axis in a single plot. 2) '2AXIS' - Two series of data are plotted into a single plot against two different axes.
KEY	'left'	Location of the data label. Valid choices are: left, right, top, bottom, outside or below. Any reasonable combination of those parameters is valid as well. The choice is passed directly to gnuplot and therefore it is important to type in lower letters!
ERROR	.T.	Include element errors into the plot.

Here we give a summary of valid choices for the SELECT parameter. We also give the valid choices for horizontal and vertical data for the specified element type.

SELECT	X	Y1 or Y2	
ALL	S	E	All ele-
	N	BETA	ments
	E	MISAL	
	PSI		
QUAD	S	FIELD	Quadru-
	N	ERRFIELD	poles
	E	LENGTH	
	PSI	PSI	
		BETA	
STRUC	S	FIELD	RF struc-
	N	ERRFIELD	tures
	E	PHASE	
	PSI	ERRPHASE	
		LENGTH	
		MISAL	
		PSI	
		BETA	
		TRAJ	
XCOR	S	FIELD	X-corre-
	N	ERRFIELD	ctors
	E	LENGTH	
	PSI	PSI	
		BETA	
YCOR	S	FIELD	Y-corre-
	N	ERRFIELD	ctors
	E	LENGTH	
	PSI	PSI	
		BETA	
BPM	S	TRAJ	BPM's
	N	DISPERSION	
	E	EMIT	
	PSI	EMITGROWTH	
		BEMIT	
		BMAG	
		LUMR	
		PSI	
		BETA	
		E	
		SIGE	
		SIGE_E	
		SIZE	
		DIV	
		MISAL	
MARKER	Z	TRAJ	Marker's
	TRAJ	DIV	

```
          |    DIV          |    EMIT          |
          |                 |    E             |
          |                 |    Q             |
----------------------------------------------------------
```

Now we explain the symbols that were used in the previous table in alphabetical order:

BEMIT Normalized RMS beam emittance for the whole beam (multi-bunch).

BETA Beta function from Twiss calculation (single-particle model).

BMAG Beta mismatch magnitude.

DIV RMS beam divergence for the whole beam.

DISPERSION Beam dispersion as measured with the last MEAS_DISP command.

E Absolute beam energy.

EMIT Normalized RMS beam emittance for a single bunch.

EMITGROWTH Growth in normalized RMS beam emittance for a single bunch.

ERRFIELD Error field value (bending field, focusing field, accelerating field, ...).

ERRPHASE Error in phase between RF and beam.

FIELD Field value (bending field, focusing field, accelerating field, ...).

LENGTH Element length.

LUMR Luminosity reduction factor (similar to inverse of effective emittance).

MISAL Offset misalignment.

N Number of element.

PHASE Phase between RF and beam.

PSI Betatron phase advance from Twiss calculation (single-particle model).

Q Charge.

S Longitudinal position along the accelerator.

SIGE Absolute RMS energy spread in single bunch.

SIGE_E Relative RMS energy spread in single bunch ($sigma_E/E$).

SIZE Absolute RMS beam size.

TRAJ Beam trajectory offsets.

PRINT_LATTICE

Prints the lattice and its parameter to unit OUTLUN. Of course, first a lattice must be defined from an input deck.

PRINT_REF

Command to save data from a selected reference into a file.

FILE	-	Output file name.
REF	0	Number of reference to save (0 = none).
SILENT	.F.	Logical flag for silent mode (no standard output).

The output file provides the following information:

```
----------------------------------------
 Column  |  Information
   Nb.   |
----------------------------------------
    1    | BPM number
    2    | Longitudinal position s [m]
    3    | X average value
    4    | Error on average
    5    | X standard deviation
    6    | X root mean squared (RMS)
    7    | Y average value
    8    | Error on average
    9    | Y standard deviation
   10    | Y root mean squared (RMS)
   11    | Number of entries
----------------------------------------
```

PRINT_TWISS

Prints the calculated Twiss parameters for electrons to unit OUTLUN.

PRINT_TWISSP

Prints the calculated Twiss parameters for positrons to unit OUTLUN.

QALIGN

Quadrupole and structure alignment routine. Assuming that all trajectory deflections arise from quadrupoles, we use the BPM readings to solve for the quadrupole offsets. Using movers we align the quads in sections, then use the RF BPM readings to align the RF structures to the beam. It is possible to iterate over the solution. In order to transport the beam from one section into the next section we use dipole correctors at or close to the common quadrupole. BEWARE: This routine assumes splitted quadrupoles with a BPM and a corrector in between (NLC case). This can be modified easily (look for use of dipole correctors).

PLANE	-	Plane (X or Y) or correction (must be set).
DOLOG	.F.	Update the logbook with tracking results.
SEED	Time	Seed of random number generator for resolution.
NSTEPS	1	Number of alignment sections in linac.
ITER	5	Number of iterations.
NORF	0	Number of iterations without structure alignment.
QSTEPS	9999	Number of quadrupoles between fixed quads.
OVERLAP	5	Number of elements in tracking overlap between sections.
BPM_RESOL	0.	Resolution of BPM's [m].
RF_RESOL	0.	Resolution of RF BPM's [m].
MOVER_RESOL	0.	Resolution of quadrupole movers [m].
WEIGHT_BPM	1.	Inverse weight on BPM's for lsq routine [m].
WEIGHT_INIT	0.	Inverse weight on initial dipole correctors.
WEIGHT_MOVER	0.	Inverse weight on the quadrupole mover change [m].

QUIT

Exit the program. This command is identical to 'STOP' and 'EXIT'.

READ

Execute a command file. This command is identical to 'EXEC' and 'LOAD'. It can only be used from the interactive command level. Do not use it within a command file! LIAR will read all commands from the specified file and switch back to the interactive program level at its end.

FILE	-	Name of input command file (must be specified).

READ_MAD

Read the lattice and the optics from a MAD TAPE input deck (compare description of input files). The initial energy must be included in the command line. In addition, one can optionaly set the number of pieces per structure, which is nominally set to 1, and the number of structures per girder, which is also nominally set to 1. If NGIRDER is specified, every NGIRDER structures are assigned to a girder. There is no check for quads or drifts that might disrupt this girder assignment.

INFILE	-	Input file name (must be set).
ENERGY	2.	Injection energy in GeV.
NPIECE	1	Number of pieces per structure.
NGIRDER	1	Number of structures per girder.

READ_SLC

Read the actual SLC settings from a file that is generated within the SLC steering panel. Such a file contains the measured trajectories, the corrector settings, the quadrupole settings and the assumed energy profile along the SLC linac. Reading such a file allows adjusting the LIAR model as closely to SLC as possible.

INFILE	-	Input file name (must be set).
ORBIT	.F.	Logical flag for reading trajectory information into a reference which is specified by the option SAVE.
SAVE	1	Specifies where the reference trajectory information is saved. Twenty different references are allowed so the option SAVE must take values between 1 and 20.
QUAD	.F.	Logical flag for reading quadrupole information and adjusting the RF acceleration to reproduce the SLC energy profile. Reading the quadrupole information replaces the original information! The acceleration is adjusted by putting RF gradient errors.
CORR	.F.	Logical flag for reading corrector information into a reference (cannot be changed).
POSITRON	.F.	Flag for electrons or positrons (not implemented).
PRINT	.F.	Print gradient errors from adjusting the energy profile into the local file 'E.PRINT'.

READ_SLC_ORBIT

Read an SLC linac orbit file into a BPM reference.

FILE	-	Input file name (must be set).
REF	0	Reference number to save the orbit in (1 to 20).
CHANGEXY	.F.	Flag for exchanging x and y information.

READ_TRNS

Read the lattice and the optics from a TRANSPORT input deck (compare description of input files). The initial energy must be included in the command line. In addition, one can optionaly set the number of pieces per structure, which is nominally set to 1, and the number of structures per girder, which is also nominally set to 1. If NGIRDER is specified, every NGIRDER structures are assigned to a girder. There is no check for quads or drifts that might disrupt this girder assignment.

INFILE	-	Input file name (must be set).
ENERGY	2.	Injection energy in GeV.
NPIECE	1	Number of pieces per structure.
NGIRDER	1	Number of structures per girder.

REF_AVG

Calculate and save the average of a number of reference trajectories.

FIRST	-	Number of first reference to include.
NB	-	Number of references to include.
SAVE	-	Number of reference where to save the average.

REF_SUB

Calculate and save the difference between two reference trajectories.

REF1	-	Number of reference 1.
REF2	-	Number of reference 2.
REF3	-	Number of reference where to save REF1-REF2.

RESET

Reset parts or all of the memory information in LIAR to zero.

ALL	.F.	Reset all information to zero.
BEAM	.F.	Reset the beam information to zero.
CONTROL	.F.	Reset the control information to zero.
RF	.F.	Reset the RF information to zero.
LOGBOOK	.F.	Reset the logbook to zero.
INITIAL	.F.	Reset the initial conditions to zero.
FEEDBACK	.F.	Reset the feedback information to zero.
LATTICE	.F.	Reset the lattice information to zero.
CORRECTOR	.F.	Reset all corrector strengths to zero.
TRAJFIT	.F.	Reset parameters for trajectory fit routines.

RFALIGN

Align the RF structures to the beam. Before this routine is called the beam must be tracked through the lattice in order to update the RF BPM's. RF BPM's are located in the beginning and the end of each RF structure. BEWARE: If the structures are divided into pieces then the RF BPM errors due to piece to piece misalignments are included. Otherwise those errors should be included in the resolution error ("effective resolution"). For realignment we use both RF BPM's per structure!

PLANE	'BOTH'	Plane of structure realignment ('X', 'Y', 'BOTH').
SIGX	0.	Horizontal resolution of RF BPM.
SIGY	0.	Vertical resolution of RF BPM.
IFIRST	1	First support structure to realign.
ILAST	1	Last support structure to realign.
SEED	Time	Seed for random number generator.
CUT	3	Cutoff in sigmas for random number distribution.
SILENT	.F.	Logical flag for silent mode (no standard output).

RF_SWITCHOFF

Switch off the accelerating RF and rescale the lattice after a specified location in the beamline. The short-range beam loading can be taken into account (compare SET_RF).

S0	0.	Location for start of switching off.
KLOSS	0.	Loss factor from beam loading in V/C/m (optional).
CHARGE	0.	Bunch charge in number of particles per bunch for beam loading (optional).
SILENT	.T.	Option for silen mode (reduced output).

SCALE_LATTICE

Scale all quadrupole and corrector strengths by a specified amount. This is equivalent to an energy change and allows to evaluate the dispersion. The scaling is done by attaching relative strength errors to all magnet strengths. The errors can then easily be reset.

ERROR	0.	Relative error for quadrupole and corrector strengths (with respect to their nominal values). If ERROR, for example, is -0.1 then all nominal fields are multiplied by 0.9.
RESET	.T.	Logical flag for resetting previous strength errors.
SILENT	.F.	Logical flag for silent mode (no standard output).

SENS_YKICK

Get sensitivity of final BPM reading for kicks along the linac (R12 elements).

DKICK	100.d-6	Strength of kick [rad] to be used.
FILE	'sens_ykick.out'	Name for output file name.
NAV	16	Number of BPM's for averaging of R12's at the end of the linac.

SENS_YQUAD

Get sensitivity of final BPM reading for quadrupole offsets along the linac (equivalent R12 elements).

DY	100.d-6	Strength of kick [rad] to be used.
FILE	'sens_ykick.out'	Name for output file name.
NAV	16	Number of BPM's for averaging of R12's at the end of the linac.

SET_BEAM

Set up the beam in bunches, slices and monoenergetic macro-particles. The command assumes that the initial conditions have been set up properly with SET_INITIAL. The default values for SET_BEAM are only set for the first call. All calls after the first call take their defaults from the last previous call. Therefore there is no need to respecify all options each time.

CURRENT	-	Number of electrons per bunch (must be set).
ERRCURRENT	0.	One σ amplitude of a random error that is added to the value of CURRENT (e.g. current jitter).
CUTCURRENT	3.	Number of sigma cut for random e^- current error.
CURRENTP	-	Number of positrons per bunch.
ERRCURRENTP	0.	One σ amplitude of a random error that is added to the value of CURRENTP.
CUTCURRENTP	3.	Number of sigma cut for random e^+ current error.
BLENGTH	-	RMS bunch length in m (must be set).
ERRBLENGTH	0.	One σ amplitude of a random error that is added to the value of BLENGTH (e.g. bunch length jitter).
CUTCURRENT	3.	Number of sigma cut for random e^- current error.
ECUT	3	Energy cut in number of sigmas.
ZCUT	3	Bunch length cut in number of sigmas.
EZCORREL	0.	Correlation between longit. position z and relative energy deviation $\Delta E/E$ of a particle: $(\Delta E + E)/E = 1 + \mathrm{EZCORREL} * z$. EZCORREL is defined in units of 1/m.
NB	1	Number of bunches (must be set).
NS	20	Number of slices (must be set).
NM	5	Number of macro-particles (must be set).
BSPACE	0.	Nominal bunch spacing in m for long-range wakefields.
POSITRON	.F.	Define first bunch as a positron bunch.
SILENT	.F.	Logical flag for silent mode of command (no standard output).
ZFILE	' '	Optional name of input file to define initial bunch distribution (integral normalized to 1). The parameters NS, ZCUT and BLENGTH are ignored and determined from the input file.

SET_BPMREF

Modify selected BPM reference information manually. Useful for example in order to specify a closed bump in a reference that then is implemented by reference steering (compare TRACKC).

NAME	' '	Name of BPM.
REF	0	Number of reference to modified.
RESET	.F.	Flag for resetting old reference value to zero.
X	0.	Reference value X in m.
Y	0.	Reference value Y in m.

SET_CHARGE

Scale the charge of a specified bunch.

BUNCH	1	Number of bunch to be changed in charge.
SCALE	1.	Scale factor for the bunch charge (1=no change).

SET_CONTROL

Set the general control parameter for LIAR.

DEBUG	0	Debug level.
OUTLUN	6	Unit number for output (6=terminal). The allowed unit numbers range from 30 to 49. You must select a unit from this range.
OUTFILE	-	Optional output file name if OUTLUN is not 6.

SET_CORRECTOR

Change the field value of a dipole corrector.

NAME	' '	Name of the dipole corrector.
RESET	.F.	Flag for resetting old field value to zero.
FIELD	0.	Field change in T*m.
KICK	0.	Alternatively, kick change in radian.

SET_FEEDBACK

Change settings of a specified feedback loop.

NAME	-	Name of feedback loop to set (must be set).
PLANE	'both'	Active plane for feedback (X,Y,BOTH,NONE).
BUNCH	1	Active bunch for feedback (0=all).
X	0.	Desired Δx in m.
XANG	0.	Desired $\Delta x'$ in rad.
Y	0.	Desired Δy in m.
YANG	0.	Desired $\Delta y'$ in rad.
RESET	.F.	Logical flag for resetting of setpoints.

SET_INITIAL

Set initial beam conditions. The command has to be called separately for electrons and positrons. The default values for SET_INITIAL are only set for the first call. All calls after the first call take their defaults from the last previous call. Therefore there is no need to respecify all options each time. After changing initial conditions the command SET_BEAM must be called so that the changes are implemented. DO NOT specify EMITX/Y and NEMITX/Y at the same time. Decide for one option!

POSITRON	.F.	Define initial conditions for positrons. Default is electrons.
X	0.	Initial x-position in m.
ANGX	0.	Initial x-divergence in rad.
Y	0.	Initial y-position in m.
ANGY	0.	Initial y-divergence in rad.
ENERGY	0.	Injection energy in GeV.
ESPREAD	0.	Initial energy spread in GeV.
EMITX	0.	Initial horizontal emittance in rad*m.
EMITY	0.	Initial vertical emittance in rad*m.
NEMITX	0.	Normalized initial horizontal emittance in rad*m.
NEMITY	0.	Normalized initial vertical emittance in rad*m.
BETAX	0.	Initial β_x in m.
BETAY	0.	Initial β_y in m.
ALPHAX	0.	Initial α_x.
ALPHAY	0.	Initial α_y.
MATCH	.F.	Match the initial β's and α's to the TWISS information (e.g. specified with command CALC_TWISS).

SET_MULTIKNOB

Change settings of a specified multiknob. The new value is always added on top of the old value.

NAME	-	Name of multiknob to set (must be set).
VALUE	0.	Change in multiknob value.

SET_RF

Set the RF phases (e.g. BNS damping). Specify only one kind of switching location (energy, element name or s-position). Up to six different RF phases can be specified. We use the following convention for RF phases: A *positive* RF phase will put the beam *behind* the crest; a *negative* RF phase will put it *before* the crest. Mathematically this means that for a particle at location z its total phase with respect to the RF on crest ($\phi_{\mathrm{RF}} = 0$ assumed to be at $z = 0$) is:

$$\phi_{\mathrm{total}} = \phi_{\mathrm{RF}} + k_{\mathrm{RF}} \cdot z \tag{103}$$

where k_{RF} is the wave number $2\pi f_{\mathrm{RF}}/c$ of the accelerating RF. Remember that $z < 0$ corresponds to a location before the crest, e.g. the head of the bunch!

NAME	'*'	Name of structures to set (must be set). '*' selects all structures. The string can be used for searching on names (e.g. 'K02*').
FRF	Lattice	RF frequency in Hz (optional).
SCALE	.F.	Flag for scaling lattice to final beam energy.
ENERGY	0.	Final beam energy in GeV.
GRAD	Lattice	Gradient in GeV/m (instead of ENERGY parameter).
PHASE1	0.	First RF phase in degree.
⋮	0.	
PHASE6	0.	Sixth RF phase in degree.
NSWITCH1	' '	Name of structure to switch phase.
LSWITCH1	INFTY	Longitudinal position of phase switch in m. There is no switch if this value is larger than the accelerator length.
ESWITCH1	INFTY	Energy of phase switch in GeV. There is no switch if this value is larger than the final beam energy.
⋮		
NSWITCH5	' '	Name of structure to switch phase.
LSWITCH5	INFTY	Longitudinal position of phase switch in m.
ESWITCH5	INFTY	Energy of phase switch in GeV.
KLOSS	0.	Loss factor from beam loading in V/C/m (optional).
CHARGE	0.	Bunch charge in number of particles per bunch for beam loading (optional).

SET_LRWF_SA

Set-up simplified long-range wakefields. Long-range wakefields from one bunch to the next are represented by a point-like wakefield kick plus its first derivative in position z along the bunch. If a structure is broken into several pieces the wakefield stays the same along the whole structure. This is in contrast to the command SET_WF_TRANSV_LR that actually assigns a single wakefield mode to each cell of the structure, thus allowing a more realistic representation of long-range wakefields in the presence of internal structure misalignments.

TFILE	-	Filename for transv. wakefield input (must be set).
LFILE	-	Filename for long. wakefield input (must be set).
TRANSV	.T.	Flag for transverse wakefields.
LONGIT	.T.	Flag for longitudinal wakefields.

SET_SR_WF

Set-up the short-range wakefields.

FILE	-	Filename for wakefield input (must be set).
TRANSV	.T.	Flag for transverse wakefields.
LONGIT	.T.	Flag for longitudinal wakefields.

SET_SRQ_WF

Set-up the short-range wakefields including quadrupole wakes. The input file contains the quadrupole wakefield information after the dipole wakefield information.

FILE	-	Filename for wakefield input (must be set).
TRANSV	.T.	Flag for transverse wakefields.
LONGIT	.T.	Flag for longitudinal wakefields.

SET_WF_TRANSV_LR

Set the long-range transverse wakefields. Assumes that the number of specified modes (see input files) is equal to the number of cells in the structure.

FILEBASE	' '	Filebase for transverse long-range wakefield file.
SEQSTART	101	Start value for file sequence number.
FILE	' '	Optional output file for wakefields including frequency errors.
GENERATE	.F.	Flag for generating LRWF frequency errors.
FREQ_ERROR	0.	RMS frequency error $\Delta f/f$ for long-range wakefields.
SEED	time	Seed for random number generator.
NUM_ERRTYPE	NUM.STRUC_ERRTYPE	Number of different errors.

Allowing for different types of structures and wakefield frequency errors we have a specific convention for the file name. Input files are

- of form *FILEBASE*,
- concatenated with '_' and a 2-digit *structure type* number,
- concatenated with '_' and a 3-digit *sequence number*,
- concatenated with '.in_tlrw'.

The 3-digit integer sequence numbers start from SEQSTART and increase in integer order up to SEQSTART + NUM.STRUC_ERRTYPE - 1. For example, if SET_WF_TRANSV_LR is called with the parameters

```
FILEBASE = '../infiles/DDS1'
SEQSTART = 101
```

and the internal variables for the number of structure types and the number of frequency errors are found to be

```
NUM.STRUC_TYPE    = 4
NUM.STRUC_ERRTYPE = 50
```

then the wake input file names are assumed to be

```
../infiles/DDS1_01_101.in_tlrw
../infiles/DDS1_01_102.in_tlrw
   ...
../infiles/DDS1_01_150.in_tlrw
   ...
../infiles/DDS1_03_101.in_tlrw
../infiles/DDS1_03_102.in_tlrw
   ...
../infiles/DDS1_03_150.in_tlrw
```

The user must make sure that all those files for different structure types and different frequency errors exist. Note that frequency errors can also be generated internally in LIAR.

SHOW_DEFINITIONS

Print the maximum dimensions of basic LIAR data structures. Those dimensions are defined in the file 'definitions.inc' and cannot be changed after compilation of LIAR.

SHOW_FEEDBACK

Print a list of all feedback loops that are defined.

SHOW_MARKER

Print a list of all MARKER points that are defined. They can be used for feedback loops and emittance bumps.

SHOW_MISALIGN

Print information about the misalignment and connected information. The standard setup prints a summary of RMS misalignments and errors to the standard output. In addition one or more beamline element types can be selected for ASCII table output into a file.

QUAD	.F.	Logical flag for output of quadrupole information.
BPM	.F.	Logical flag for output of BPM information.
RF	.F.	Logical flag for output of RF structure information.
BEND	.F.	Logical flag for output of bending magnet information.
XCOR	.F.	Logical flag for output of X corrector information.
YCOR	.F.	Logical flag for output of Y corrector information.
FILE	-	Generic filename for output. The extension is added depending on the beamline element type.
BUNCH	1	Bunch number for BPM reading output.

The files contain the following informations (for every structure piece one obtains one line of output):

```
--------------------------------------------------------------------
  Element type    |    File    |    Column    |    Information
                  | extension  |    Number    |
-----------------|-----------|------------|--------------------------
                  |            |              |
  BPM             |    .BPM    |       1      | BPM number
                  |            |       2      | S-position [m]
                  |            |       3      | X misalignment [m]
                  |            |       4      | Y misalignment [m]
                  |            |       5      | X BPM reading [m]
                  |            |       6      | Y BPM reading [m]
                  |            |       7      | X beam position [m]
                  |            |       8      | Y beam position [m]
                  |            |       9      | X emittance [m-rad]
                  |            |      10      | Y emittance [m-rad]
                  |            |      11      | X luminosity reduction
```

		12	Y luminosity reduction
Quadrupoles	.QUA	1	Quadrupole number
		2	S-position [m]
		3	Rel. strength error
		4	X misalignment [m]
		5	Y misalignment [m]
		6	Tilt [rad]
		7	Integrated field [Tm]
RF / struct.	.RFU	1	Structure number
		2	S-position [m]
		3	Phase error [rad]
		4	Rel. gradient error
		5	X piece misalignment [m]
		6	Y piece misalignment [m]
		7	X of first RF BPM [m]
		8	X of second RF BPM [m]
		9	Y of first RF BPM [m]
		10	Y of second RF BPM [m]
X / Y corr.	.XCR	1	Corrector number
	.YCR	2	S-position [m]
		3	Rel. field error
		4	Tilt [rad]
		5	Integrated field [Tm]

SHOW_MULTIKNOB

Print a list of all multiknobs that are defined.

SHOW_SUPPORT

Print information about the accelerator support structure.

STOP

Exit the program. This command is identical to 'QUIT' and 'EXIT'.

TRACK

Track the beam through the lattice. The summary output at the end of the tracking is explained in Table I.

ORDER	2	Order of beam transport (1=centroid, 2=beam ellipse). Full beam ellipse (second order) tracking should be used in most cases. Chromaticity and dispersion are always included to all orders.
COUPLE	.T.	Flag for coupled beam transport. If false all element tilts (e.g. skew quadrupoles) are ignored.
WF_T_SR	.T.	Flag for transverse short-range wakefields.
WF_T_LR	.T.	Flag for transverse long-range wakefields.
WF_L_SR	.T.	Flag for longitudinal short-range wakefields.
WF_L_LR	.T.	Flag for longitudinal long-range wakefields.
WF_Q_SR	.F.	Flag for transverse, short-range quadrupole wakefields.
E_AXIS	.F.	Flag for emittance calculation wrt axis instead of the centroid position. Used for emittance analysis and saving of emittance at BPM's et cetera.
E_GEOM	.F.	Flag for geometric emittance instead of normalized one. Used for emittance analysis and saving of emittance at BPM's et cetera.
INIT	.T.	Flag for reinitialization of beam to initial conditions. Normally true, because e.g. several seeds are calculated for the same accelerator, always resetting the beam to the same starting point. If false, multitracking can be realized. For example a beam could be tracked through N identical FODO cells by tracking it N times through a single FODO cell. However, the use of this option seems to be limited.
MBUNCH	1	Bunch selection for detailed slice information.
RFBUNCH	1	Bunch selection for RF BPM.
DOLOG	.F.	Flag for saving end results into logbook.
MEAS_SLICE_BUNCH	0	Save the trajectories of all slices along the beamline into the file 'slice.bpm'. Saved are BPM-Nr, S, XBEAM, YBEAM, X(slice1), Y(slice1), ..., X(sliceN), Y(sliceN). No measurement for MEAS_SLICE_BUNCH equal 0.

TRACKC

Track the beam through the lattice with 1-to-1 correction on the fly. For additional details see TRACK!

ORDER	2	Order of beam transport (1=centroid, 2=beam ellipse).
COUPLE	.T.	Flag for coupled beam transport.
WF_T_SR	.T.	Flag for transverse short-range wakefields.
WF_T_LR	.T.	Flag for transverse long-range wakefields.
WF_L_SR	.T.	Flag for longitudinal short-range wakefields.
WF_L_LR	.T.	Flag for longitudinal long-range wakefields.
WF_Q_SR	.F.	Flag for transverse, short-range quadrupole wakefields.
E_AXIS	.F.	Flag for emittance calculation wrt axis instead of the centroid.
E_GEOM	.F.	Flag for geometric emittance instead of normalized one.
INIT	.T.	Flag for reinitialization of beam to initial conditions.
MBUNCH	1	Bunch selection for detailed slice information.
RFBUNCH	1	Bunch selection for RF BPM.
DOLOG	.F.	Flag for saving end results into logbook.
PLANE	'both'	Plane for 1-to-1 correction ('X','Y','BOTH').
BUNCH	0	Selection of bunch for correction (0=all).
CORS	'standard'	Selection of corrector scheme ('STANDARD', 'SLC').
XFILE	-	X-Corrector, BPM association file for option 'SLC'.
YFILE	-	Y-Corrector, BPM association file for option 'SLC'.
BPMX	'qf'	Selection of BPM's for horizontal plane ('QF', 'QD', 'ALL','*').
BPMY	'qd'	Selection of BPM's for vertical plane ('QF', 'QD', 'ALL', '*').
MINPHASE	$\pi/4$	Minimum phase advance between corrector and associated BPM.
MAXPHASE	$3\pi/4$	Maximum phase advance between corrector and associated BPM.
ITER	1	Number of iterations.
BEGIN	first	Number of first element for correction.
END	last	Number of last element for correction.
RESOL	0.	RMS BPM resolution in m.
CUT	3	Sigma cutoff for BPM resolution errors.
REF	0	If REF is not equal zero then the steering will be done to the BPM reference with number 'REF'.

The standard correction scheme uses correctors at QF's for the horizontal correction and correctors at QD's for the vertical correction. Correction schemes may fail due to specifics in the used lattice. A new lattice generally has to be made working with this routine! BPM's can be specified with a search string, e.g. BPMY = 'MY*' will select all BPM's with names that start with MY.

TRACKCP

Track the beam through the lattice with 1-to-1 correction assuming positively charged particles. Same options as in command TRACKC but the roles of focusing and defocusing magnets are exchanged.

TRACKF

Track the beam through the lattice and adjust feedbacks. For additional details see TRACK!

ORDER	2	Order of beam transport (1=centroid, 2=beam ellipse).
COUPLE	.T.	Flag for coupled beam transport.
WF_T_SR	.T.	Flag for transverse short-range wakefields.
WF_T_LR	.T.	Flag for transverse long-range wakefields.
WF_L_SR	.T.	Flag for longitudinal short-range wakefields.
WF_L_LR	.T.	Flag for longitudinal long-range wakefields.
WF_Q_SR	.F.	Flag for transverse, short-range quadrupole wakefields.
E_AXIS	.F.	Flag for emittance calculation wrt axis instead of the centroid.
E_GEOM	.F.	Flag for geometric emittance instead of normalized one.
INIT	.T.	Flag for reinitialization of beam to initial conditions.
MBUNCH	1	Bunch selection for detailed slice information.
RFBUNCH	1	Bunch selection for RF BPM.
ITER	3	Number of iterations per feedback.
DOLOG	.F.	Flag for saving end results into logbook.
SILENT	.T.	Flag for silent mode (no standard output).

TRACKFC

Track the beam through the lattice, adjust feedbacks and do a 1-to-1 correction on the fly after feedback loops that have zero setpoints. For additional details see TRACK and TRACKC!

ORDER	2	Order of beam transport (1=centroid, 2=beam ellipse).
COUPLE	.F.	Flag for coupled beam transport.
WF_T_SR	.T.	Flag for transverse short-range wakefields.
WF_T_LR	.T.	Flag for transverse long-range wakefields.
WF_L_SR	.T.	Flag for longitudinal short-range wakefields.
WF_L_LR	.T.	Flag for longitudinal long-range wakefields.
WF_Q_SR	.F.	Flag for transverse, short-range quadrupole wakefields.
E_AXIS	.F.	Flag for emittance calculation wrt axis instead of the centroid.
E_GEOM	.F.	Flag for geometric emittance instead of normalized one.
MBUNCH	1	Bunch selection for markers.
ITER	3	Number of iterations per feedback.
DOLOG	.F.	Flag for saving end results into logbook.
MBUNCH	1	Bunch selection for detailed slice information.
RFBUNCH	1	Bunch selection for RF BPM.
CORS	'standard'	Selection of corrector scheme ('STANDARD', 'SLC').
XFILE	-	X-Corrector, BPM association file for option 'SLC'.
YFILE	-	Y-Corrector, BPM association file for option 'SLC'.
BPMX	'qf'	Selection of BPM's for horizontal plane ('QF', 'QD', 'ALL', '*').
BPMY	'qd'	Selection of BPM's for vertical plane ('QF', 'QD', 'ALL', '*').
MINPHASE	$\pi/4$	Minimum phase advance between corrector and associated BPM.
MAXPHASE	$3\pi/4$	Maximum phase advance between corrector and associated BPM.
CITER	1	Number of iterations for correction.
SILENT	.F.	Flag for silent mode (no standard output).

The standard correction scheme uses correctors at QF's for the horizontal correction and correctors at QD's for the vertical correction. Correction schemes may fail due to specifics in the used lattice. A new lattice generally has to be made working with this routine! BPM's can be specified with a search string, e.g. BPMY = 'MY*' will select all BPM's with names that start with MY.

TRAJFIT

Fit the trajectory offset and divergence at a location with NBPM downstream BPM readings. The fit is calculated analytically and is done for both planes. Results are printed to standard output. An internal version of this routine is available, passing the results as arguments.

ELEM0	-	Number of element where x, x', y and y' are to be fitted. Either ELEM0 or NAME must be specified.
NAME	-	Name of element where x, x', y and y' are to be fitted. Either ELEM0 or NAME must be specified.
NBPM	8	Number of downstream BPM's to be included in the fit.
BUNCH	0	Number of bunch that is used for the fit (0 = all).
REFNUM	0	Number of reference that is subtracted from the BPM information (0 = none).
ERROR	.T.	Logical flag to include BPM offset errors.
WFUDGE	1.	Possible wakefield fudge factor to adjust for defocussing effect of wakefields (1. = no wakefields).
SILENT	.F.	Logical flag for silent mode (less standard output).

7 DO-IT-YOURSELF EXTENSIONS TO LIAR: THE INTERNAL DATA STRUCTURE

We describe here how the user can add its own subroutine. The internal structure of LIAR is explained in more detail. The implementation of the command language allows easy use of subroutines without knowing details about their content.

7.1 Understanding the command language

As explained before the command language accepts commands of the following form:

```
MISALIGN_QUAD, DX = 5.E-06, DY = 7.E-06, TILT = 1.E-06,
RESET = .T., NAME = 'Q02*';
```

This command would randomly misalign all quadrupoles in the beamline. From the command line LIAR extracts the following informations:

```
1.  Subroutine to be called:               MISALIGN_QUAD
2.  Number of REAL*8 parameters:           3
3.  List of REAL*8 par. names (CHAR*8):    ('DX', 'DY', 'TILT')
4.  List of par. values (REAL*8):          (5.E-06, 7.E-06, 1.E-06)
5.  Number of LOGICAL parameters:          1
6.  List of LOGICAL par. names (CHAR*8):   ('RESET')
7.  List of par. values (LOGICAL):         (.true.)
8.  Number of STRING parameters:           1
9.  List of STRING par. names (CHAR*8):    ('NAME')
10. List of par. values (STRING):          ('Q02*')
```

The main program of LIAR will read a command line, get the subroutine name (= command name) and call it passing the number of parameters, a list of parameter names and a list of parameter values over a COMMON block. As parameter values we foresee REAL*8, LOGICAL's and STRING's at the moment. If necessary the REAL*8 values can be converted into INTEGER values inside the called subroutine ("ROUND()" utility function).

7.2 Adding a command and subroutine

For adding a subroutine first copy the *liar* directory tree to your local account (see above). Then go into the *src_v1.9* subdirectory and do all your changes there. You should

1. add your subroutine to the file *addition.f* and
2. specify the command and subroutine name to the main program in the file *liar.f*. Just look for

   ```
   c---------------------------------------------------
   c++     A D D    C O M M A N D S    I N    H E R E:
   c---------------------------------------------------
   ```

   then copy an existing 'ELSEIF' block and modify the command name, the subroutine name and the comment as appropiate for your addition.

DO NOT MODIFY EXISTING SUBROUTINES. COPY THEM INTO addition.f AND RENAME THEM! ONLY AFTER THAT YOU SHOULD MODIFY THEM. MODIFYING ORIGINAL COMMANDS/SUBROUTINES WILL LIKELY RESULT IN SERIOUS INCOMPATIBILITIES!

It is important to note that all subroutines are called with an errorflag (INTEGER*4) as the only parameter:

```
INTEGER*4     ERRFLAG
...
CALL  subroutine(ERRFLAG)
IF (ERRFLAG.NE.0) THEN
  WRITE(*,*) 'ERR>  Command failed: ', PAR.COMMAND
ENDIF
```

The *Makefile* will automatically detect your addition and both compile and link it. You need to type

```
make liar
```

on the UNIX level, while in the *src_v1.9* directory. A new executable 'liar' will be generated. Just add it into directory *bin.aix6000*. The command can now be used inside LIAR.

After the addition is tested and debugged it can be included into the general release of the program by submitting it to the LIAR's coordinator. A short description of the command and its parameters should be supplied as well.

7.3 Accessing information from other parts of LIAR

Any added subroutine needs to access internal information from other parts of LIAR. All important informations in LIAR are organized in FORTRAN *records*. Records are an extended FORTRAN feature that is available for most FORTRAN compilers. Records have the power to combine components of different types into a single structure. This will become clear later. The records are defined in include files and put into COMMON blocks. They can then be accessed by including the proper include files, e.g.:

```
INCLUDE 'definitions.inc'  ! Dimensional constants
INCLUDE 'par.inc'          ! Command parameters
INCLUDE 'control.inc'      ! Control parameters
INCLUDE 'constants.inc'    ! Physics constants
INCLUDE 'lattice.inc'      ! Lattice description
INCLUDE 'initial.inc'      ! Initial conditions
INCLUDE 'beam.inc'         ! Beam description
INCLUDE 'logbook.inc'      ! Logbook data
INCLUDE 'feedback.inc'     ! Feedback loops and emittance bumps
```

From the documentation of the include files on the following pages it is seen how all kinds of specific informations are accessed. Note that 'dimensions.inc' ALWAYS must be included as the first include file.

7.4 Dimensional constants and definitions

All basic dimensional constants are defined in the include file 'definitions.inc'. It MUST be included as the FIRST include file! Manipulating the dimensions in this file allows to adjust LIAR to a particular problem, thus avoiding memory problems. The dimensions are transferred to the subroutine by including the file:

```
dimensions.inc
```

The dimensional constants can be accessed from the subroutine (e.g. for checking of dimensions) in the following way:

NBUNCH_MAX	Maximum number of bunches.
NB_MAX	
NSLICE_MAX	Maximum number of slices per bunch.
NSMAX	
NS_MAX	
NMP_MAX	Maximum number of monoenergetic beam ellipses per slice.
NBEAMP_MAX	Maximum number of beam particles. This is NBUNCH_MAX * NSLICE_MAX * NMP_MAX.
STRUC_TYPE_MAX	Maximum number of different RF structure types.
WF_ERROR_MAX	Maximum number of error types for transverse long-range wakefields.
NPIECE_MAX	Maximum number of pieces per RF structure.
NPIECE_PER_STRUC_MAX	
NFDBK_MAX	Maximum number of feedbacks.
NEBUMPMAX	Maximum number of emittance bumps.
NELEM_MAX	Maximum number of beamline elements.
NQUAD_MAX	Maximum number of quadrupoles.
NBEND_MAX	Maximum number of bending magnets.
NBPM_MAX	Maximum number of BPM's.
NSTRUCT_MAX	Maximum number of RF structures.
NXCOR_MAX	Maximum number of horizontal dipole correctors.
NYCOR_MAX	Maximum number of vertical dipole correctors.
NMARKER_MAX	Maximum number of marker points.
MAX_NREF	Maximum number of reference points per BPM.
NSUPPORT_MAX	Maximum number of support girders.
NATTACH_MAX	Maximum number of elements that are attached to a single girder.
NRFBPM_MAX	Maximum number of RF BPM's per structure.
NSEED_MAX	Maximum number of seeds per set in the logbook.
NSET_MAX	Maximum number of sets in the logbook.
NPAR_MAX	Maximum number of parameters for a single LIAR command.
NMULTI_MAX	Maximum number of multi-knobs.
NMULTI_ELEMENT_MAX	Maximum number of elements per multi-knob.

7.5 Command parameters

A command is defined with any number of options. The options are transferred to the subroutine in the record *PAR* by including the file:

```
par.inc
```

The parameters can be accessed from the subroutine in the following way:

PAR.COMMAND	Command and subroutine name.
PAR.RNUM	Number of REAL*8 parameters specified with command.
PAR.RNAME(i)	Name of i^{th} parameter specified.
PAR.RVALUE(i)	Value of i^{th} parameter specified (REAL*8).
PAR.LNUM	Number of LOGICAL parameters specified with command.
PAR.LNAME(i)	Name of i^{th} parameter specified.
PAR.LVALUE(i)	Value of i^{th} parameter specified (LOGICAL).
PAR.SNUM	Number of STRING parameters specified with command.
PAR.SNAME(i)	Name of i^{th} parameter specified.
PAR.SVALUE(i)	Value of i^{th} parameter specified (STRING).

The following example illustrates how the REAL*8 parameters could be processed in a subroutine (all variables execept 'PAR.*' are defined locally). LOGICAL and STRING parameters are accessed in equivalent loops.

```
      REAL*8       DX, DY, TILT, XCUT, YCUT, TILTCUT
      INTEGER*4    SEED
c
c++   Set default RMS values for hor. and vert. misalignment, tilt and
c++   cut parameters (multiples of DX, DY, TILT):
c
      DX      = 0.E-06
      DY      = 0.E-06
      TILT    = 0.E-06
      XCUT    = 2.5
      YCUT    = 2.5
      TILTCUT = 2.5
      SEED    = 0
c
c++   Read the REAL*8 command parameters and change selected parameters to
c++   specified values:
c
      DO i = 1, PAR.RNUM
        IF (PAR.RNAME(i).EQ.'DX') THEN
          DX      = PAR.RVALUE(i)
        ELSEIF (PAR.RNAME(i).EQ.'DY') THEN
          DY      = PAR.RVALUE(i)
        ELSEIF (PAR.RNAME(i).EQ.'TILT') THEN
          TILT    = PAR.RVALUE(i)
        ELSEIF (PAR.RNAME(i).EQ.'XCUT') THEN
          XCUT    = PAR.RVALUE(i)
        ELSEIF (PAR.RNAME(i).EQ.'YCUT') THEN
          YCUT    = PAR.RVALUE(i)
        ELSEIF (PAR.RNAME(i).EQ.'TILTCUT') THEN
          TILTCUT = PAR.RVALUE(i)
        ELSEIF (PAR.RNAME(i).EQ.'SEED') THEN
          SEED    = ROUND(PAR.RVALUE(i))
        ELSE
          WRITE(*,*) 'ERROR> Parameter not recognized: ', PAR.RNAME(i)
        ENDIF
```

```
END DO
```

This approach allows a large flexibility in adding subroutines and command parameters. If a parameter is not specified in the command line its default value is used.

7.6 Control parameters

The central control for the debug level and the default output unit is done with the include file

`control.inc`

which defined the variables:

DEBUG	Debug level.
OUTLUN	Default output unit (screen, file, ...).

Those variables should consistently be used to control the debug and the standard output. Local varibales with the same name must not be defined!

7.7 Physics constants

The include file

`constants.inc`

defines physical constants that are useful in many parts of the program:

CONST_VC	Speed of light in m/s.
CONST_QE	Electron charge in Coulomb.
CONST_PI	Value of π.
CONST_EMASS	Mass of the electron in GeV.
CONST_INFTY	Infinity approximation (10^{30}).
CONST_BCONV	Conversion factor in $p = reB$.

Local varibales with the same name must not be defined!

7.8 Lattice description

As all the other information the lattice is saved in the form of FORTRAN records, which are defined as structures. We define a list of elements which guides the user through the lattice. The actual elements are saved in structures according to their types (quads, structures, ...). This allows an easy and intuitive access to the properties of each element. By including the file

`lattice.inc`

the following information can be accessed and changed.

Element list The beamline is described as a list of elements. An element holds all information which is not specific to the type of the element. For specific information it points to the adequate entry in a type specific list.

ELEMENT(i).NAME	Name of element i (up to eight characters).
ELEMENT(i).S	Longitudinal position of head of element i in m.
ELEMENT(i).DS	Longitudinal misalignment of quadrupole i in m. This should not change the order of elements!
ELEMENT(i).LDRIFT	Length of drift before head of element in m.
ELEMENT(i).ENERGY	Design energy at element i in GeV.
ELEMENT(i).IS_QUAD	Logical flag whether element i is a quad or not.
ELEMENT(i).IS_BEND	Logical flag whether element i is a bending magnet or not.
ELEMENT(i).IS_STRUC	Logical flag whether element i is a structure or not.
ELEMENT(i).IS_BPM	Logical flag whether element i is a BPM or not.
ELEMENT(i).IS_XCOR	Logical flag whether element i is a X-corrector or not.
ELEMENT(i).IS_YCOR	Logical flag whether element i is a Y-corrector or not.
ELEMENT(i).IS_MARKER	Logical flag whether element i is a MARKER or not.
ELEMENT(i).POINTER	Position number of element i in type specific list of quad's, structures, ...

Example:

```
IF (ELEMENT(i).IS_QUAD) WRITE(*,*) QUAD(ELEMENT(i).POINTER).K
```

Quadrupole list All quadrupoles share the same basic structure. Splitted quadrupoles are allowed! Elements which are slices of one physical quadrupole should all point to the same entry in this list (assuming same misalignments).

QUAD(i).LENGTH	Length of quadrupole i in m.
QUAD(i).K	Integrated nominal quadrupole strength in T.
QUAD(i).DK	Relative error in quadrupole strength ($\Delta K/K$).
QUAD(i).DX	Horizontal misalignment of quadrupole i in m.
QUAD(i).DY	Vertical misalignment of quadrupole i in m.
QUAD(i).TILT	Rotation about the longitudinal direction in Radian.

Bending magnet list Bending magnets are implemented without synchrotron radiation!

BEND(i).LENGTH	Length of quadrupole i in m.
BEND(i).B	Nominal bending field in T.
BEND(i).DB	Relative error in bending field ($\Delta B/B$).
BEND(i).TILT	Rotation about the longitudinal direction in Radian.
BEND(i).EDGE1	
BEND(i).EDGE2	

BPM list All BPM's share the same basic structure. During each tracking the beam positions and emittances for all bunches are saved.

BPM(i).XBEAM(j)	Horizontal position of beam in BPM i in m (saved during last tracking) for all bunches j (wrt axis x=0).
BPM(i).YBEAM(j)	Vertical position of beam in BPM i in m (saved during last tracking) for all bunches j (wrt axis y=0).
BPM(i).XEMIT(j)	Horizontal bunch emittance (saved during last tracking) for all bunches j.
BPM(i).YEMIT(j)	Vertical bunch emittance (saved during last tracking) for all bunches j.
BPM(i).XEMITALL	Horizontal beam emittance (saved during last tracking).
BPM(i).YEMITALL	Vertical beam emittance (saved during last tracking).
BPM(i).XLUMF	Average X luminosity reduction for bunch train.
BPM(i).YLUMF	Average Y luminosity reduction for bunch train.
BPM(i).XBMAG(j)	Horizontal BMAG (saved during last tracking) for all bunches j.
BPM(i).YBMAG(j)	Vertical BMAG (saved during last tracking) for all bunches j.
BPM(i).DX	Horizontal misalignment of BPM i in m.
BPM(i).DY	Vertical misalignment of BPM i in m.
BPM(i).ETAX	Horizontal dispersion at BPM i in m.
BPM(i).ETAY	Vertical dispersion at BPM i in m.
BPM(i).TILT	Tilt of BPM i in rad.
BPM(i).RESOL	RMS resolution of BPM i in m.
BPM(i).XREF(k,2)	Value of X-reference nb. k at BPM number i. The two fields allow for example to store the sum and the sum of the squares.
BPM(i).YREF(k,2)	Value of Y-reference nb. k at BPM number i. The two fields allow for example to store the sum and the sum of the squares.
BPM(i).NREF(k)	Number of entries for the reference nb. k. Allows to calculate the mean, RMS, error on the mean.
BPM(i).REFSTAT(k)	Status of BPM number i for reference nb. k (0=bad, 1=good).
BPM(i).STATUS	Status of the actual BPM inside LIAR (0=bad, 1=good).
BPM(i).ENERGY	Energy in GeV of the tracked beam at the actual BPM.
BPM(i).ESPREAD	Beam energy spread in GeV of the tracked beam.
BPM(i).XSIZE	Horizontal beam size during last tracking in m.
BPM(i).YSIZE	Vertical beam size during last tracking in m.
BPM(i).XPSIZE	Horizontal divergence during last tracking in rad.
BPM(i).YPSIZE	Vertical divergence during last tracking in rad.
REF_BPMY(k)	Type of reference number k (Character*16).

RF Structure list RF structures are defined as physical entities allowing separate misalignments for any number of sub-pieces ("ipiece").

STRUC(i).LENGTH	Length of structure piece i in m.
STRUC(i).RFPHASE	Nominal phase of RF with respect to head of first bunch in Radian.
STRUC(i).ERRPHASE	Error of RF phase (Δf_{RF}) in Radian.
STRUC(i).GRADIENT	Nominal gradient of acceleration in GeV/m.
STRUC(i).ERRGRAD	Relative error of accelerating gradient ($\Delta E_{grad}/E_{grad}$).
STRUC(i).DX(ipiece)	Horizontal misalignment of structure piece i in m.
STRUC(i).DY(ipiece)	Vertical misalignment of structure piece i in m.
STRUC(i).TYPE	INTEGER number indicating the structure type (e.g. SLC has 3 different structure types, dimpling).
STRUC(i).ERRTYPE	The modes in each structure can have frequency errors. Only a limited number of such frequency errors is actually generated and randomly assigned to structures. This INTEGER number gives the error type for each structure.
STRUC(i).RFBPM_X(2)	Horizontal up- and downstream beam position (wrt x=0) if the RF structure i is used as an RF BPM.
STRUC(i).RFBPM_Y(2)	Vertical up- and downstream beam position (wrt y=0) if the RF structure i is used as an RF BPM.
STRUC(i).RESOL	Resolution of RF BPM.
STRUC(i).FRF	RF frequency of particular structure.

Horizontal corrector list The horizontal correctors are described by the following information:

XCOR(i).LENGTH	Length of horizontal corrector i in m.
XCOR(i).FIELD	Integrated dipole field of horizontal corrector i in Tesla·m.
XCOR(i).DFIELD	Relative field error ($\Delta B/B$).
XCOR(i).TILT	Rotation about longitudinal axis in Radian.
XCOR(i).REF	Reference corrector strength.

Vertical corrector list The vertical correctors are described by the following information:

YCOR(i).LENGTH	Length of vertical corrector i in m.
YCOR(i).FIELD	Integrated dipole field of vertical corrector i in Tesla·m.
YCOR(i).DFIELD	Relative field error ($\Delta B/B$).
YCOR(i).TILT	Rotation about longitudinal axis in Radian.
YCOR(i).REF	Reference corrector strength.

Marker list We save the average beam position and average emittance at the locations of markers at the time of the tracking:

MARKER(i).XEMITALL	Horizontal beam emittance.
MARKER(i).YEMITALL	Vertical beam emittance.
MARKER(i).XSIZE	Horizontal beam size in m.
MARKER(i).YSIZE	Vertical beam size in m.
MARKER(i).XPSIZE	Horizontal divergence in rad.
MARKER(i).YPSIZE	Vertical divergence in rad.
MARKER(i).XBEAM(j)	Horizontal position in m for all bunches j.
MARKER(i).YBEAM(j)	Vertical position in m for all bunches j.
MARKER(i).XANGBEAM(j)	Horizontal divergence in rad for all bunches j.
MARKER(i).YANGBEAM(j)	Vertical divergence in rad for all bunches j.
MARKER(i).XEMITBEAM(j)	Horizontal emittance for all bunches j.
MARKER(i).YEMITBEAM(j)	Vertical emittance for all bunches j.
MARKER(i).ENERGYBEAM(j)	Energy in GeV for all bunches j.
MARKER(i).QBEAM(j)	Charge in C for all bunches j.
MARKER(i).XLUMF(j)	X luminosity reduction for all bunches j.
MARKER(i).YLUMF(j)	Y luminosity reduction for all bunches j.
MARKER(i).X(k)	Horizontal position in m for all slices k.
MARKER(i).Y(k)	Vertical position in m for all slices k.
MARKER(i).XANG(k)	Horizontal divergence in rad for all slices k.
MARKER(i).YANG(k)	Vertical divergence in rad for all slices k.
MARKER(i).XEMIT(k)	Horizontal emittance for all slices k.
MARKER(i).YEMIT(k)	Vertical emittance for all slices k.
MARKER(i).ENERGY(k)	Energy in GeV for all slices k.
MARKER(i).Q(k)	Charge for all slices k.

The slice informations are saved only for the bunch specified by the parameter “MBUNCH” during the last tracking.

List of relevant numbers In order to use the information above one needs to know the number of elements, quad's, . . . :

NUM.ELEM	Total number of elements in lattice.
NUM.QUAD	Total number of quadrupoles.
NUM.PIECE_PER_STRUC	Number of pieces each accelerating structure is divided into.
NUM.PIECE	Total number of pieces of accelerating structures.
NUM.STRUC	Total number of structures.
NUM.SUPPORT	Total number of support structures (girders).
NUM.GIRDER	Total number of girders (obsolete).
NUM.STRUC_TYPE	Number of different structures in lattice.
NUM.STRUC_ERRTYPE	Number of structure types with different frequency errors which are randomly assigned to the structures.
NUM.BPM	Total number of BPM's.
NUM.BEND	Total number of bending magnets.
NUM.XCOR	Total number of horizontal correctors.
NUM.YCOR	Total number of vertical correctors.
NUM.MARKER	Total number of markers.
NUM.SUPPORT	Total number of support structures defined.

Twiss parameters When calculated the e- Twiss parameters are saved into the following structure for all elements:

TWISS(i).PSIX	Horiz. phase advance in rad for all elements i.
TWISS(i).PSIY	Vert. phase advance in rad for all elements i.
TWISS(i).BETAX	Horiz. β function in m for all elements i.
TWISS(i).BETAY	Vert. β function in m for all elements i.
TWISS(i).ALPHAX	Horiz. α for all elements i.
TWISS(i).ALPHAY	Vert. α for all elements i.

For positrons the TWISS parameters are saved into TWISSP.

Accelerator support The information about the accelerator support is saved in the data structure SUPPORT:

SUPPORT(i).FOR_QUAD	Logical flag for quadrupole support (only one support point).
SUPPORT(i).FOR_STRUC	Logical flag for RF structure support (two support points).
SUPPORT(i).POINTER(j)	Pointer to every element that is attached to the support structure i.
SUPPORT(i).S1	Lonitudinal position of first support point [m].
SUPPORT(i).S2	Lonitudinal position of second support point [m].
SUPPORT(i).NATTACH	Number of attached elements for support structure i.

For positrons the TWISS parameters are saved into TWISSP.

7.9 Initial conditions

The initial conditions are defined with the command 'SET_INITIAL' which reads the specified parameters and fills the common block accordingly. By including the file

```
initial.inc
```

the following information can be accessed and changed.

INIT.X	Initial horizontal position in mm.
INIT.ANGX	Initial horizontal angle in mrad.
INIT.Y	Initial vertical position in mm.
INIT.ANGY	Initial vertical angle in mrad.
INIT.ENERGY	Initial averaged beam energy in GeV.
INIT.ESPREAD	Initial energy spread σ_E in GeV.
INIT.EMITX	Initial emittance in radian·m.
INIT.EMITY	Initial emittance in radian·m.
INIT.NEMITX	Normalized initial emittance in radian·m.
INIT.NEMITY	Normalized initial emittance in radian·m.
INIT.BETAX	Initial beta function in m.
INIT.BETAY	Initial beta function in m.
INIT.ALPHAX	Initial α.
INIT.ALPHAY	Initial α.

The structure INIT refers to electrons. If positrons are considered as well the initial positron information is stored in INITP. Initial conditions can be defined before or after building the lattice. They must be set before setting up the beam (command 'SET_BEAM'). If they are changed (incoming jitter) the 'SET_BEAM' command must be rerun. Example of use:

```
INCLUDE 'definitions.inc'
INCLUDE 'initial.inc'
...
EMIT0_ELEC = INIT.NEMITY
EMIT0_POSI = INITP.NEMITY
```

7.10 Beam and wakefield description

The beam is divided into bunches that are divided into slices that are divided into monoenergetic macro-particles. The bunches are allowed to have an arbitrary 3-dimensional charge distribution. According to the distribution the charge is distributed to all slices and macro-particles. All bunches have the same charge distribution.

We assume that the wavelength of all long-range wakefields is long compared to the bunch length. With this assumption it is sufficient to calculate the long-range wakefields for every bunch distance. By supplying their slopes with respect to z the strengths of wakefields can be extrapolated within a bunch (assuming only one weighted average frequency). With the slopes we can also allow for small errors in bunch spacing.

By including the file

```
/afs/slac.stanford.edu/public/software/liar/release/src_v1.9/beam.inc
```

the following information can be accessed and changed.

Beam information The beam is divided into bunches, slices and macro-particles. Since we might track up to 100,000 beam particles (or beam ellipses in second order transport) the handling of this information limits the CPU performance. We try to implement the beam in a way that maximizes the cache hit rate. For each beam particle we have phase space and beam size (sigma matrix) information. The sigma matrix is symmetric so that only a triangle has to be saved.

BEAM(i).CHARGE — Charge associated with beam particle i in C.

BEAM(i).X(j) — Phase space vector of beam particle i. “j” runs from 1 to 6 with:

j = 1	is	x	(horizontal position in m)
2	is	x'	(horizontal beam angle in radian)
3	is	y	(vertical position in m)
4	is	y'	(horizontal beam angle in radian)
5	is	z	(long. pos. wrt center of bunch in m)
6	is	E	(beam energy in GeV)

BEAM(i).SIGMA(j) — Sigma matrix for the 2-dimensional coupled case and beam particle i. The sigma matrix carries the beam size information. “j” runs from 1 to 10 with:

j = 1	is	SIGMA(1,1)
2	is	SIGMA(2,1)
3	is	SIGMA(2,2)
4	is	SIGMA(3,1)
5	is	SIGMA(3,2)
6	is	SIGMA(3,3)
7	is	SIGMA(4,1)
8	is	SIGMA(4,2)
9	is	SIGMA(4,3)
10	is	SIGMA(4,4)

The beam information is a big memory block (4 MByte with 100 bunches, 30 slices and 10 macro-particles). Its size determines also the speed of the tracking algorithm.

In order to have maximum calculation speed (no blank fields in data) we need to use a **strict convention**. Beam particles always MUST be accessed in the following way:

```
ICOUNT = 0
DO IBUNCH = 1, NBUNCH
  DO ISLICE = 1, NSLICE
    DO IMP = 1, NMP
      ICOUNT = ICOUNT + 1
      BEAM(ICOUNT).CHARGE = QPART
      BEAM(ICOUNT).X(6) = ENERGY
    END DO
  END DO
```

```
END DO
```

An INTEGER*4 function "IBEAM(ibunch, islice, imp)" is provided in order to directly get the position of a beam particle in the one-dimensional beam array.

Beam information Beam informations (actual numbers of bunches, slices, macro-particles, bunch length) are accessed over:

NR.BUNCH	Actual number of bunches in beam.
NR.SLICE	Actual number of slices per bunch.
NR.MP	Actual number of macro-particles per slice.
NR.BEAMPART	Actual number of beam particles (= NR.BUNCH * NR.SLICE * NR.MP).
NR.SLICESPACING	Nominal distance between adjacent slices in m.
NR.BUNCHLENGTH	Nominal length of single bunch in m.
NR.BUNCHSPACING	Nominal distance between adjacent bunches in m.

Transverse dipole wakefields Transverse dipole wakefields are saved for each possible difference in slice/bunch position ("$\Delta_{\text{slice, bunch}}$"). We always assume that the relative slice positions are the same in all bunches. The structures are symmetric so that the wakefields are the same in both transverse directions. If structures are broken into pieces the wakefield kicks need to be specified for each piece position ("ipiece"). For transverse wakefields the convolution with the beam must be done for both transverse planes before tracking through each beamline piece (the kick depends on actual transverse beam positions).

WF_T.SR(Δ_{slice})	Transverse kick in V/C/m^2 from short-range transverse wakefields for each difference Δ_{slice} in slice position. Short-range wakefields decay long before the next bunch arrives.
WF_T.LR(Δ_{bunch}, ipiece, itype, ierr)	Transverse kick in V/C/m^2 from long-range transverse wakefields for each difference in bunch positions (Δ_{bunch}), each piece position ("ipiece") and each frequency error type ("ierr").
WF_T.DLR(Δ_{bunch}, ipiece, itype, ierr)	Slope of transverse long-range wakefields wrt z in V/C/m^3 (assuming only one weighted average frequency). This allows both to extrapolate the long-range wakefields within a bunch and to include small errors in bunch spacing.
WF_T.DDLR(Δ_{bunch}, ipiece, itype, ierr)	Second derivative of transverse long-range wakefields wrt z in V/C/m^4.
WF_T.THETAX(ibunch, islice)	Horizontal kick from transverse wakefields for slice 'islice' in bunch 'ibunch'.
WF_T.THETAY(ibunch, islice)	Vertical kick from transverse wakefields for slice 'islice' in bunch 'ibunch'.

Transverse quadrupole wakefields Transverse short-range quadrupole wakefields are saved for each possible difference in slice position (“Δ_{slice}”). We always assume that the relative slice positions are the same in all bunches. The structures are symmetric so that the wakefields are the same in both transverse directions.

WF_Q.SR(Δ_{slice})	Transverse kick in V/C/m^2 from short-range transverse quadrupole wakefields for each difference Δ_{slice} in slice position. Short-range wakefields decay long before the next bunch arrives.
WF_Q.THETAX(ibunch, islice)	Horizontal kick from transverse quadrupole wakefields for slice 'islice' in bunch 'ibunch'.
WF_Q.THETAY(ibunch, islice)	Vertical kick from transverse quadrupole wakefields for slice 'islice' in bunch 'ibunch'.
WF_Q.R(i, j, ibunch, islice)	Quadrupole wakefield R-matrix for slice 'islice' in bunch 'ibunch'. Both i and j run from 1 to 5.

Longitudinal wakefields Longitudinal wakefields are saved for each possible difference in slice/bunch position (“$\Delta_{\text{slice, bunch}}$”). We always assume that the relative slice positions are the same in all bunches. If structures are broken into pieces the wakefield kicks need to be specified for each piece position (“ipiece”). For longitudinal wakefields the convolution with the beam is done only once for each beam setup. Allowing for intensity jitter it is done once at the beginning of each tracking.

WF_L.SR(Δ_{slice})	Deacceleration in V/C/m from short-range longitudinal wakefields for each relative slice position.
WF_L.LR(Δ_{bunch}, ipiece)	Deacceleration in V/C/m from long-range longitudinal wakefields for each bunch spacing and each piece position (“ipiece”). The compensation with the RF voltage must be taken into account correctly! We assume no errors in the accelerating RF voltage.
WF_L.DLR(Δ_{bunch}, ipiece)	Slope of longitudinal long-range wakefields wrt z in V/C/m^3 (assuming only one weighted average frequency). This allows both to extrapolate the long-range wakefields within a bunch and to include small errors in bunch spacing.
WF_L.DDLR(Δ_{bunch}, ipiece)	Second derivative of longitudinal long-range wakefields wrt z in V/C/m^4.
WF_L.ELOSS(ibunch, islice, ipiece)	Longitudinal wakefield deacceleration after convolution with beam in [GeV] for every bunch/slice, each piece position (“ipiece”).

7.11 Feedback loops and emittance bumps

In LIAR feedback loops and emittance bumps can be implemented at marker points. The include file

`feedback.inc`

provides access to the internal definitions of feedback loops and emittance bumps.

A feedback loop is implemented such that the observation point is set to any marker. The two most efficient upstream dipole correctors are found in order to manipulate x, x', y and y' at the observation marker point. All changes are done with respect to reference values. The following information is defined for the feedback loops:

IFDBK	Actual number of defined feedbacks.
FDBK(i).NAME	Name of feedback loop i.
FDBK(i).IMARKER	Pointer to relevant observation marker.
FDBK(i).IELEM_M	Element number of relevant marker.
FDBK(i).BUNCH	Active bunch for feedback observation (0=all).
FDBK(i).PLANE	Active plane for feedback (X,Y,BOTH,NONE).
FDBK(i).REF_X	Reference x-offset at observation point.
FDBK(i).REF_XANG	Reference x-divergence at observation point.
FDBK(i).REF_Y	Reference y-offset at observation point.
FDBK(i).REF_YANG	Reference y-divergence at observation point.
FDBK(i).X	Desired Δx at observation point.
FDBK(i).XANG	Desired $\Delta x'$ at observation point.
FDBK(i).Y	Desired Δy at observation point.
FDBK(i).YANG	Desired $\Delta y'$ at observation point.
FDBK(i).ICOR1X	Pointer to x-corrector nb. 1.
FDBK(i).IELEM_C1X	Element number of x-corrector nb. 1.
FDBK(i).ICOR2X	Pointer to x-corrector nb. 2.
FDBK(i).IELEM_C2X	Element number of x-corrector nb. 2.
FDBK(i).C1_R12X	R12 from x-corrector 1 to marker point.
FDBK(i).C1_R22X	R22 from x-corrector 1 to marker point.
FDBK(i).C2_R12X	R12 from x-corrector 2 to marker point.
FDBK(i).C2_R22X	R22 from x-corrector 2 to marker point.
FDBK(i).ICOR1Y	Pointer to y-corrector nb. 1.
FDBK(i).IELEM_C1Y	Element number of y-corrector nb. 1.
FDBK(i).ICOR2Y	Pointer to y-corrector nb. 2.
FDBK(i).IELEM_C2Y	Element number of y-corrector nb. 2.
FDBK(i).C1_R12Y	R12 from y-corrector 1 to marker point.
FDBK(i).C1_R22Y	R22 from y-corrector 1 to marker point.
FDBK(i).C2_R12Y	R12 from y-corrector 2 to marker point.
FDBK(i).C2_R22Y	R22 from y-corrector 2 to marker point.

Since the internal information of the emittance bumps is likely not to be needed by many people it is not described here. It can be looked up in the include file.

7.12 Logbook data

A logbook structure is defined for keeping results in memory. It is managed with pre-existing commands (see above) and should not necessarily be modified from other subroutines. It is accessed with the include file

`logbook.inc`

For more details please look into this include file.

8 DESCRIPTION OF INPUT FILES

In order to run LIAR a number of input files needs to be supplied. The standard files can be found in

```
/afs/slac.stanford.edu/public/software/liar/release/infiles
```

Available standard files for LIAR version 1.9 at this moment include:

```
1.  slc.trns                SLC design lattice (TRANSPORT deck)
2.  nlcII.trns              NLCIIb design lattice (TRANSPORT deck)
3.  pre_linac.tape          NLC pre-linac design lattice (MAD file)
4.  srwf_nlc.dat            Short-range wakefield file for NLC
5.  srwf_slc.dat            Short-range wakefield file for SLC
6.  srwf_q_slc.dat          Short-range wakefield file for SLC with
                               quadrupole wakefields
7.  lrwf_t_slc.dat          Long-range transverse wakefield for SLC.
8.  lrwf_l_slc.dat          Long-range longitudinal wakefield for SLC.
9.  DDSTE10_01_001.in_tlrw  Long-range transv. wakefield file for NLC
10. lat90.trns              90 deg lattice at 20 GeV with structures
                               but without nominal acceleration.
```

Special files needed for some LIAR options are:

```
1. xmatch.ext_elec     SLC X-corrector - BPM matching file
2. ymatch.ext_elec     SLC Y-corrector - BPM matching file
3. slc.trns_new        SLC design lattice with splitted RF structures
4. ldp_format          SLC phase advance format file
5. sigz_e42_slc.dat    Measured longitudinal bunch distribution
                       (available for 30 MV to 42 MV).
```

In the next sections we give descriptions and references for the input formats which are used by LIAR. If LIAR is to be extended to other accelerators than SLC and NLC those formats should be used or alternatively new input routines can be written.

8.1 Lattice description

The lattice can be entered as either a MAD TAPE file or a TRANSPORT input file. The MAD TAPE files are generated using the MAD program and the TWISS command. The format of these files is described in the MAD manual. The TRANSPORT file format must be the 'old' numeric format where element types are identified by numeric codes[9]. Labels are enclosed in single or double quotes and up to eight characters are read. Continuation lines are distinguished with ambersands and comments begin with an opening parathesis and extend for the rest of the line. Units can be changed with type code 15 but are NOT set with the commands 'UMAD' or 'UTRANS'. A list of recognized lattice elements and their TRANSPORT codes is given in Table II. Note that bending magnets are not treated correctly in LIAR version 1.9. This is an example of part of an SLC input file:

```
  5.      'Q110401T'    0.05340    57.11113   106800.0          ;
 13.      'M110401T'  503.                                      ;
  5.      'Q110401T'    0.05340    57.11113   106800.0          ;
  3.                    0.02700                                 ;
 11.      'K1100411'    0.32680     0.006069       0.0  10.5    ;
  7.01    'X110402T'                                            ;
```

[9]Note that all specified numbers in the TRANSPORT deck are assumed to be real numbers. Integer numbers are not allowed.

Type	Label(1:1)	Element	Param. 1	Param. 2	Param. 3	Param. 4
3.	-	DRIFT	Length [m]	-	-	-
4.	-	BEND	Length [m]	B [kG]	-	-
5.	-	QUAD	Length [m]	B [kG]	Aperture [cm]	-
7.	X	XCOR	-	-	-	-
	Y	YCOR	-	-	-	-
11.	-	STRUC	Length [m]	Δ E [GeV]	ϕ [degree]	λ_{RF} [cm]
13.	M	BPM	-	-	-	-
	X	XCOR	-	-	-	-
	Y	YCOR	-	-	-	-
	(else)	MARKER	-	-	-	-
15.	-	UNIT	Unit nb.	Scale factor	-	-
16.	-	S0	6.	S0 [cm]	-	-
20.	-	TILT	α [degree]	-	-	-

Table II: Definitions and conventions for TRANSPORT input decks that are recognized by LIAR. Some element types use a specific convention for labelling. Note that bending magnets are not yet implemented correctly in version 1.9. Note further, that parameters which have no importance for some elements (indicated by '-') may be omitted. Any specified types and parameters MUST have type REAL (e.g. do use '0.' instead of '0' !. Units can be changed with type '15.'. For unit numbers see Table III.

```
 11.      'K1100412'    0.25000     0.004643       0.0  10.5     ;
  7.0001  'Y110403T'                                             ;
 11.      'K1100413'    1.60481     0.029803       0.0  10.5     ;
  3.                    0.86249                                  ;
 11.      'K1100414'    2.18161     0.040514       0.0  10.5     ;
  3.                    0.86249                                  ;
 11.      'K1100415'    3.04410     0.056531       0.0  10.5     ;
 11.      'K1100416'    2.18161     0.040514       0.0  10.5     ;
  3.                    0.66269                                  ;
(Comment: Here is a MARKER for a wire scanner
 13.      'W110444T'  504.                                       ;
  3.                    0.23400                                  ;
  5.      'Q110501T'    0.05340   -57.70276   106800.0           ;
 13.      'M110501T'  503.                                       ;
  5.      'Q110501T'    0.05340   -57.70276   106800.0           ;
```

Note that the type '7.' here is refined into types '7.01' and '7.0001'. LIAR only considers the INTEGER part of types and does not use those extensions! The corrector type is specified by the first letter of its label ('X' or 'Y')!

For both TRANSPORT and MAD input decks, the initial energy must be included in the LIAR command line. In addition, one can optionaly set the number of pieces per structure, which is nominally set to 1, and the number of structures per girder, which is also nominally set to 1. If NGIRDER is specified, every NGIRDER structures are assigned to a girder. There is no check for quads or drifts that might disrupt this girder assignment.

Unit	Description	Default
1.	Aperture	cm
5.	Wavelength λ	cm
7.	Angle	degree
8.	Length	m
9.	Field	kG
11.	Energy	GeV

Table III: Unit numbers and default units for LIAR TRANSPORT input decks. Units can be changed with the transport type ' 15.' !

8.2 Wakefields

The wakefields are not calculated inside LIAR. It is assumed that the wakefield calculation is performed separately from the beam dynamics calculations with specialized programs. The wakefields are specified to LIAR in the form of tables. LIAR then calculates the convolution of the wakefields with the beam. In the following we describe the wakefield tables:

8.2.1 Short-range transverse and longitudinal wakefields

Short-range wakefields are set from the command SET_SR_WF. The short range wakefields are specified by one wakefield input file, which gives both the longitudinal (monopole) and the transverse (dipole) wakefields of cylindrically symmetric accelerating structures. These two functions are sufficient for obtaining the behavior of an arbitrarily shaped, ultra-relativistic, short bunch in the linac, if the transverse beam size is small compared to the structure iris radius a, if the orbit excursions are $r < a/2$, and if the beta function is large compared to the structure period. The longitudinal wakefields give the average longitudinal voltage loss, per meter of structure, experienced by a test particle a distance s behind an ultra-ralativistic point driving charge of charge 1 pico-coulomb. The transverse wakefields give the average transverse voltage gain, per meter of structure and per meter of offset of the driving charge r_d, experienced by a test particle a distance s behind the same point driving charge.

The wakefield input file is expected to begin with several lines of comments (indicated by a '(' in the first row), followed by a number n of records of longitudinal data, with each record giving an index (not used), an s position (in meters), and the wakefield (in V/pC/m). The parameter s should begin at zero and increase in equal increments. The program reads this data and uses cubic spline interpolation to find the wakes at intermediate points. The largest values of s needs to be sufficiently large to accommodate the largest separation of two macro-particles in the bunch to be tracked. Next in the file come more comment cards followed again by n records of transverse data, with each record giving an index (not used), an s position (the same set as before), and the wakefield (in V/pC/m^2). The longitudinal coordinates for transverse and longitudinal wakefield points must agree for each point $i = 1, n$. This is an example of an input file for SLC:

```
(M=0 SLAC #45 DELTA FUNCTION WAKE)
(I  Z[M] WAKE[V/PC/M])
 1  0.000000 224.909975
 2  0.000150 180.920263
...
199  0.029700  -9.337811
200  0.029850  -9.395827
 (M=1 SLAC #45 DELTA FUNCTION WAKE)
 (I  Z[M] WAKET[V/PC/M^2])
1    0.000000      0.0000
```

```
2    0.000150    371.1632
...
199    0.029700   -227.3678
200    0.029850   -246.5704
```

For the actual calculation of short-range wakefield functions compare [8] and [7].

8.2.2 Accurate long-range transverse wakefields

We give here the format of the input file for the long range transverse wakefields (command SET_WF_TRANSV_LR). The long range transverse wakefield is of the form

$$W_{\perp}(z) = \sum_{n=1}^{N_{\mathrm{modes}}} W_n \sin(2\pi f_n z/c) \exp(2\pi f_n z/2Q_n c) \tag{104}$$

and has the units [V/C/m^2].

The format of the file is as follows:

Lines 1-4: First there are four lines allocated for descriptive information about the structure whose modes are described by the file. The character string comprising each line may be up to 80 characters long and should be enclosed in quotes.

Line 5: Number of transverse modes, N_{modes}. Is assumed to be the same as the number of cells.

Lines 6-7: Two more strings of up to 80-characters, enclosed in quotes. Typically these are used to remind the user of the format and units of the lines to follow, which contain the information about each mode.

Line 8 - (N_{modes}+7): Line for each transverse mode, containing: (1) mode number, sequential from 1 to N_{modes}, (2) mode frequency f_n in Hz, (3) coefficient of mode, W_n in [V/C/m^2], (4) location (cell number) of centroid of mode, (5) quality factor Q_n of mode.

This is an example of such an input file:

```
'DESCRIPTION: DDS, TE10 manifold, no freq errs, unif coupling, periodicity'
'COPPER Q (QCU) =   6500.00000000000000                                  '
'LEGEND29 = 7/ 6/1994  12:21  (NO ERRORS)                                '
'LEGEND25 = (PERT: Tue Sep 19 14:26:08 1995                              '
             206
'  M      FREQUENCY    2*KICK FACTOR    MODE_CEN      Q '
'            (HZ)        (V/C/M**2)                     '
     1   .12530989D+11   .46482276D+14       3   .6499996D+04
     2   .12771960D+11   .14977500D+14       6   .6499996D+04
     3   .12908022D+11   .10556679D+14       8   .6499996D+04
     4   .13008189D+11   .86671965D+13      10   .6499996D+04
     5   .13087500D+11   .76209984D+13      11   .6499996D+04
     6   .13150877D+11   .69696504D+13      13   .6499996D+04

    ........

   202   .15954405D+11   .82348892D+14     201   .8058751D+03
   203   .16014419D+11   .63097705D+14     202   .1089590D+04
   204   .16087801D+11   .53580452D+14     203   .1611691D+04
   205   .16186417D+11   .53811225D+14     204   .2269957D+04
   206   .16358843D+11   .10981940D+15     206   .2107941D+04
```

For the actual calculation of the wakefield function for SLAC examples compare [10].

8.2.3 Simplified long-range transverse wakefields

Transverse long-range wakefields can be specified in a simplified form with the command SET_LRWF_SA (only for two bunches right now). Wakefields from one bunch to the next are represented by a point-like wakefield kick $V/pC/m^2$ plus its first derivative in position z along the bunch. If a structure is broken into several pieces the wakefield stays the same along the whole structure. This is in contrast to the command SET_WF_TRANSV_LR that actually assigns a single wakefield mode to each cell of the structure, thus allowing a more realistic representation of long-range wakefields in the presence of internal structure misalignments. However, for many purposes the simplified treatment is adequate.

The input file is organized into comment lines (indicated by a '(' in the first row) and data records that consist 1) of the longitudinal position s [m] behind the leading bunch and 2) the wakefield in $V/pC/m^2$. We give an example of an input file:

```
(SLC TRANSVERSE WAKE; TRSL451
(S[M   ] WAKE[V/PC/meter/meter]
0.      0.
0.001   2541.54
0.002   3128.86
0.003   3710.29
0.004   4082.
0.005   4327.14
...
0.496   -365.829
0.497   -154.343
0.498   116.049
0.499   394.657
```

8.2.4 Long-range longitudinal wakefields

Long-range longitudinal wakefields are only implemented in the simplified form as described in the previous section. At this time only a two bunch scheme is implemented, but a scheme for n bunches can easily be provided.

The input file is organized into comment lines (indicated by a '(' in the first row) and data records that consist 1) of the longitudinal position s [m] behind the leading bunch and 2) the longitudinal wakefield in $V/pC/m$. We give an example of an input file:

```
(SLC LONGITUDINAL WAKE
(S[M   ] WAKE[V/PC/M])
 0.0000E+00   2.5850E+02
 7.5000E-04   1.2527E+02
 1.5000E-03   9.1271E+01
 2.2500E-03   7.1900E+01
 3.0000E-03   5.8458E+01
...
 2.9625E-01  -1.3539E+00
 2.9700E-01  -5.4442E+00
 2.9775E-01  -1.6743E+01
 2.9850E-01  -1.5822E+01
 2.9925E-01  -5.6519E+00
```

8.2.5 Arbitrary initial bunch distribution along z

The initial bunch distribution in z can be defined from an input file. Its integral is required to be 1. An example for SLC (compare 'infiles' directory is:

```
#
#  SLC bunch shape as measured with the Streak camera
#
#
#  Particle             :  e-
#  Compressor voltage   :  42 MV
#
#  * The data is averaged over 30-40 measurements
#  * Time jitter was removed
#  * Integral of distribution normalized to 1
#
#  Data: R. Holtzapple (reformatted: R. Assmann)
#
#
# Time [ps]       z [mm]      Fract. Intensity
#
  21.00000      6.295642      .1013589E-03
  20.00000      5.995850      .1749323E-03
  19.00000      5.696057      .5837724E-03
  18.00000      5.396265      .4360712E-03
  17.00000      5.096472      .7381052E-03
  16.00000      4.796680      .8068226E-03
  15.00000      4.496888      .6950982E-03
  14.00000      4.197095      .2000588E-02
  13.00000      3.897302      .1352341E-02
  12.00000      3.597510      .1416716E-02
  11.00000      3.297718      .1661893E-02
  10.00000      2.997925      .3258084E-02
  9.000000      2.698133      .6305721E-02
  8.000000      2.398340      .1494190E-01
  7.000000      2.098547      .2751879E-01
  6.000000      1.798755      .4453740E-01
  5.000000      1.498963      .5440400E-01
  4.000000      1.199170      .6634378E-01
  3.000000      .8993775      .7310159E-01
  2.000000      .5995850      .7526019E-01
  1.000000      .2997925      .8234391E-01
  .0000000E+00  .0000000E+00  .8815681E-01
 -1.000000     -.2997925      .9749197E-01
 -2.000000     -.5995850      .9986860E-01
 -3.000000     -.8993775      .9490541E-01
 -4.000000     -1.199170      .7611185E-01
 -5.000000     -1.498963      .4963293E-01
 -6.000000     -1.798755      .2484623E-01
 -7.000000     -2.098547      .1100318E-01
```

The number of specified slices must not exceed the maximum number of slices as defined in the LIAR include files.

9 LIST OF PROGRAM SUBROUTINES AND FUNCTIONS

Here we provide alphabetical lists of subroutines and functions. We also specify the file where they are located. The main routine is located in the file 'liar.f'. We also list the FORTRAN unit file numbers used in the program. Generally numbers can be used by different subroutines as long as the files are closed at the end of the subroutine. Note that the unit number 77 is reserved for the LIAR command file and must not be used by any subroutine!

1. Subroutines are defined in the following files:

Subroutine	File name
ADJUST_LATTICE	utilities.f
ATL	align_support.f
ATLMOVE	align_support.f
AUTOPHASE	align_support.f
CHECK_LATTICE	read_lat.f
CALC_LUMF	track.f
CALC_LUMF2	track.f
CALC_TWISS	read_lat.f
CALC_TWISSP	read_lat.f
CLOSE_FILE	utilities.f
CONV_QUADW	quadwake.f
CONV_LONG	track.f
CONV_TRANSV	track.f
CORR_BPM	dfs.f
CORRECT	correction.f
CORRECT_QUAD	correct_quad.f
CORRECT$QUAD	correct_quad.f
CORRECT$QUAD2	align_support_suppl.f
CORRECTP	correction.f
COVAR	correction.f
CSPLIN	set_sr_wf.f
DEFINE_CORR	correction.f
DEFINE_FEEDBACK	correction.f
DEFINE_MULTIKNOB	multi.f
DEFINE_SUPPORT	align_support.f
DFSX	dfs.f
DFSY	dfs.f
DLUM_WAIST	utilities.f
DOTRAJFIT	phase.f
DUMMY	addition.f
EMIT2	emit.f
EMITC	correction.f
EMITC_DEF	correction.f
EMITC_DEF_MKB	emitc_mkb.f
EMITC_MKB	emitc_mkb.f
ERROR_GAUSS_BPM	error_gauss.f
ERROR_GAUSS_QUAD	error_gauss.f
ERROR_GAUSS_STRUC	error_gauss.f
ERROR_HANDLE	liar.f
EXPAND	liar.f
FDBK_GOLD	align_support.f
FDBK_MISAL	align_support.f

GET_2ND	quadwake.f
GET_ESPREAD	track.f
GET_ESPREAD2	track.f
GET_POS	track.f
GET_POS2	track.f
GNUPLOT	graphics.f
HECOMP	correction.f
HOLVE	correction.f
LINEFIT	align_support.f
LINLSQ	correction.f
LOAD	liar.f
LOGBOOK	track.f
MAT_MULT	read_lat.f
MDLERR	utilities.f
MEAS_BPM	track.f
MEAS_BUNCH	track.f
MEAS_DISPERSION	dfs.f
MEAS_EMIT	track.f
MEAS_PHASE	phase.f
MEAS_PHASE2	phase.f
MEAS_REF	utilities.f
MEAS_TRAIN	track.f
MISALIGN_SUPPORT	align_support.f
NO_BLANKS	liar.f
OPEN_FILE	utilities.f
PARSE	liar.f
PARSE_TRNS	read_lat.f
PRINT_LATTICE	read_lat.f
PRINT_REF	utilities.f
PRINT_TWISS	read_lat.f
PRINT_TWISSP	read_lat.f
PROCESS_PAR	liar.f
QALIGN	align_support_suppl.f
RAN_INIT	rand_win.f / rand_aix.f
READ_MAD	read_lat.f
READ_SLC	slcref.f
READ_SLC_ORBIT	slcref.f
READ_TRNS	read_lat.f
REF_AVG	slcref.f
REF_PRINT	slcref.f
REF_SUB	slcref.f
RESET	liar.f
RESET_PAR	liar.f
RESID	correction.f
RF_SWITCHOFF	utilities.f
RFALIGN	align_support.f
RMAT_BEND	track.f
RMAT_DRIFT	track.f
RMAT_QUAD	track.f
RMAT_STRUC1	track.f
RMAT_STRUC2	track.f
RMAT_XCOR	track.f
RMAT_YCOR	track.f
RTRACK	track.f
RTRACKC	correction_suppl.f
SCALE_LATTICE	dfs.f

```
SENS_YQUAD           |   utilities.f
SENS_YKICK           |   utilities.f
SET_BEAM             |   set_beam.f
SET_BPMREF           |   utilities.f
SET_CHARGE           |   set_beam.f
SET_CONTROL          |   liar.f
SET_CORRECTOR        |   utilities.f
SET_INITIAL          |   set_initial.f
SET_FEEDBACK         |   correction.f
SET_LRWF_SA          |   wakesteer.f
SET_MULTIKNOB        |   multi.f
SET_MULTIKNOB_DO     |   multi.f
SET_MULTI_ELEMENT    |   multi.f
SET_RF               |   read_lat.f
SET_SR_WF            |   set_sr_wf.f
SET_SRQ_WF           |   quadwake.f
SET_WF_LONG_LR       |   set_wf_long_lr.f
SET_WF_TRANSV_LR     |   set_wf_transv_lr.f
SHIFT_UP             |   track.f
SHIFT_DOWN           |   track.f
SHOW_DEFINITIONS     |   utilities.f
SHOW_FEEDBACK        |   correction.f
SHOW_MARKER          |   correction.f
SHOW_MISALIGN        |   align_support.f
SHOW_MULTIKNOB       |   multi.f
SHOW_SUPPORT         |   align_support.f
TRACK                |   track.f
TRACKANA             |   trackana.f
TRACKC               |   correction.f
TRACKF               |   correction.f
TRACKFC              |   correction_suppl.f
TRACK_CENTR          |   track.f
TRACK_SIGMA          |   track.f
TRAJFIT              !   utilities.f
TRAJFIT_INTERN       |   utilities.f
UPPER_CASE           |   liar.f
----------------------------------------------
```

2. Functions are defined in the following files:

```
-------------------------------------------------
     Function        |       File name
-------------------------------------------------
COMPAR               |   correction.f
DLUM                 |   utilities.f
EMIT                 |   emit.f
EXPHASE              |   track.f
FIND_BPM             |   read_lat.f
FIND_XCOR            |   read_lat.f
FIND_YCOR            |   read_lat.f
FIND_QUAD            |   read_lat.f
FSWAKE               |   set_sr_wf.f
GAUSS_INT            |   set_beam.f
GETLEN               |   liar.f
IBEAM                |   ibeam.f
```

```
MCLOCK_LIAR            |   rand_aix.f / rand_win.f
PHASEX                 |   phase.f
PHASEY                 |   phase.f
RAN_EXP                |   rand_aix.f / rand_win.f
RAN_FLAT               |   rand_aix.f / rand_win.f
RAN_GAUSS              |   rand_aix.f / rand_win.f
ROUND                  |   liar.f
------------------------------------------------
```

3. Internal file unit numbers:

```
---------------------------------------------
    Unit number        |       File name
---------------------------------------------
        10             |   set_sr_wf.f
        11             |   slcref.f
        15             |   read_lat.f
                       |   set_beam.f
        16             |   correction.f
                       |   read_lat.f
                       |   slcref.f
        17             |   correction.f
        20             |   set_wf_transv_lr.f
        21             |   set_wf_transv_lr.f
        28             |   liar.f
        51             |   align_support.f
                       |   dfs.f
        52             |   align_support.f
        53             |   align_support.f
        54             |   align_support.f
        55             |   align_support.f
        56             |   align_support.f
        72             |   align_support.f
        73             |   dfs.f
                       |   graphics.f
                       |   logbook.f
                       |   phase.f
                       |   slcref.f
                       |   track.f
  ***   77   ***       |   liar.f    (reserved)
        80             |   phase.f
                       |   track.f
        81             |   phase.f
                       |   track.f
---------------------------------------------
```

10 WHAT' S NEW

Starting with version 1.8 we keep a list of major new features and improvements. New commands are listed in parenthesis and upper case.

10.1 Version 1.8

- Graphical output (PLOT). Ralph A.
- Multi-knob definition and emittance opyimization (DEFINE_MULTIKNOB, SET_MULTIKNOB, SHOW_MULTIKNOB, DEF_EMITC_MKB, EMITC_MKB). Tor R.
- Quadrupole wakefields (SET_SRQ_WF). Frank Z.
- Scale charge of selected bunches (SET_CHARGE). Ralph A.
- Internal generation of LRWF frequency errors. Tor R.
- Dimensional definitions moved into 'definitions.inc'. (SHOW_DIMENSIONS) Ralph A.
- Routines for manipulating BPM references and dipole corrector fields (SET_BPMREF, SET_CORRECTOR). Ralph A.
- Modified SET_SR_WF to read in any number of input lines from wakefield file. Tor R.
- Routine to define simplified transverse long-range wakefields for SLC (SET_LRWF_SA). Gennadi S.
- Mathematica interface to LIAR. Allows to use Mathematica's power to analyze and process simulation results. Gennadi S.
- Added option MEAS_SLICE_BUNCH to command TRACK in order to save all slice trajectories into output ASCII file for a selected bunch. Ralph A.
- Fixed bug in tracking routine. The transfer matrix for RF structures became ill-defined for a coasting beam (no change of beam energy $\rightarrow$ no acceleration and no beam loading!). Ralph A.
- Fixed tracking and correction for beams with bunches of opposite charges (electrons and positrons). Ralph A.

10.2 Version 1.9

- Fixed bug in energy scaling for splitted quads. This bug significantly affected the chromatic emittance growth with splitted quadrupoles. Ralph A.
- Fixed minor bugs in EXPAND and DEFINE_CORR routines. Tor R./Ralph A.
- Implemented kick angle for SET_CORRECTOR routine. Ralph A.
- Implemented MEAS_PHASE2 command. Ralph A.

11 KNOWN PROBLEMS AND BUGS

This is a list of known problems and bugs:

- Twiss calculation: The calculation of the Twiss parameters (command CALC_TWISS) is done using an approximation of high energy. For low energy cases and a large acceleration the calculation provides unprecise results. The beta function shows an apparent “beating”. The tracking results are NOT affected by this. They are correct at any energy. The problem will be fixed.

- Bending magnets are not treated correctly. Bunch compressors cannot be simulated with LIAR. A correct treatment of bending magnets will be implemented.

- UNIX: The end-of-tracking analysis prints an initial energy spread of zero though it was set to non-zero value in SET_BEAM. This problem does not affect the tracking results.

12 ACKNOWLEDGEMENTS

We wish to thank Vinod Bharadwaj and Mark Woodley for their help in checking LIAR and their many useful comments.

13 REFERENCES

[1] K. L. Brown et al., “TRANSPORT - A computer program for designing charged particle beam transport systems”. May 1977. SLAC-91, Rev. 2.

[2] K. Bane, SLAC.

[3] D.C. Carey, K.L. Brown and F.C. Iselin, “Decay Turtle (Trace Unlimited Rays Through Lumped Elements): A Computer Program for Simulating Charged Particle Beam Transport Systems, Including Decay Calculations.”, SLAC-0246, 1982.

[4] K. Kubo et al., “Phase space simulation program for main linacs of future linear colliders”. May 1995. NLC-Note 14.

[5] T. Raubenheimer, “Generation and Acceleration of Low Emittance Flat Beams for Future Linear Colliders”. PhD-thesis, SLAC-387.

[6] H. Grote and F.C. Iselin, “The MAD program version 8.10. User's reference manual”, CERN/SL/90-13(AP).

[7] F. Ebeling et al., “MAFIA user manual”, 1992.

[8] K. Bane and P. Wilson, Proc. of the 11^{th} Int. Conf. on High Energy Accelerators, CERN (Birhäuser Verlag, Basel, 1980), p. 592.

[9] B.A. Baklakov et al., “Investigation of Seismic Vibrations for Linear Collider VLEPP”. Sov Phys ZhTF, vol. 63, No. 10, 1993, p. 122 (in Russian). See also V. Shiltsev et al., “Measurements of Ground Vibrations and Orbit Motion at HERA”, DESY HERA 95-06.

[10] N.Kroll, et.al., ”Manifold damping of the NLC detuned accelerating structure”, presented at the 6^{th} Workshop on Advanced Accelerator Concepts, Lake Geneva, WI, June 12-18, 1994. SLAC-PUB-6660.

[11] A.W. Chao and R.W. Cooper, ”Transverse Quadrupole Wake Field Effects in High Intensity Linacs”, Part. Acc. Vol. 13, p. 1–12 (1983).

[12] K. Bane and P. Wilson, private communication.

[13] T.O. Raubenheimer and R.D. Ruth, “A Dispersion-Free Trajectory Correction Technique for Linear Colliders”. Nucl. Instr. and Meth. A302(1991)191.

[14] W. Spence. Unpublished note, August 1995.

[15] See one of the several home pages of the “gnuplot” project. For example:
`http://www.nas.nasa.gov/ woo/gnuplot_info.html`