BNL-220734-2020-TECH
C-A/AP/637

Transformation of Spinors in Accelerators and Beam Transfer Lines

N. Tsoupas

December 2020

Collider Accelerator Department
Brookhaven National Laboratory

U.S. Department of Energy
USDOE Office of Science (SC), Nuclear Physics (NP) (SC-26)

DISCLAIMER

Transformation of Spinors in Accelerators and Beam Transfer Lines

N. Tsoupas, F. Méot, H. Huang

Abstract

The stable spin direction and the spin tune are two important concepts used when polarized beams are transferred along beam lines or circulating in an accelerator. One of the methods to calculate the stable spin direction and the spin tune of the reference particle of a polarized beam bunch is by integrating simultaneously [1,2,3] the Lorentz Force $\vec{F} = q(\vec{E} + \vec{\upsilon} \times \vec{B})$ and the BMT equation [4] of the reference particle as it is transported by the magnets of a beam line or accelerator. A modified version of the Raytrace code [14] was used to integrate the Lorentz equation and the BMT equation. Using the spin motion which is governed by the BMT equation one can compute both, the stable spin direction of the reference particle at any point along the reference orbit of an accelerator or a beam line, and the spin tune of the closed orbit in an accelerator. The simultaneous integration of the Lorentz force and of the BMT equation is the most accurate method to calculate the trajectory and the spin motion of a charged particle moving in an electromagnetic field, however this method requires the accurate description of the electromagnetic fields of the elements in which the charged particles are moving. An alternative method of calculating the stable spin direction and the spin tune of spin ½ charged particles along the reference trajectory of a beam line or an accelerator is the transformation of the spinor which is the wavefunction of spin ½ particles. This method requires the knowledge of the stable spin direction of each element of the beam line or the accelerator. This technical note describes the "spinor-method" to calculate the stable-spin-direction and the spin-tune of a spin ½ particle moving along the reference trajectory of a beam line or an accelerator.

Introduction

The spin of the electron was hypothesized to explain the appearance of specific spectral lines in the spectrum of hydrogen atom. Such lines are the hydrogen fine structure lines and the "spin orbit interactions" [5] lines. The hypothesis of spin explains also the observations of the Stern Gerlach experiment [6] which also leads to the concept of spin quantization. The electron is considered of possessing a magnetic moment which is associated with a non-classical property of the electron called "spin". The spin was introduced originally as an ad-hoc physical quantity until P. Dirac in 1926 developed the relativistic Schrodinger's equation (Dirac's equation) [7] whose solutions were interpreted that the electrons possessed this physical quantity of spin. It was Dirac who showed that spin arises naturally in a correct relativistic formulation of the quantum theory. This formulation is embodied in the relativistic generalization of the Schrödinger equation called the *Dirac equation*.

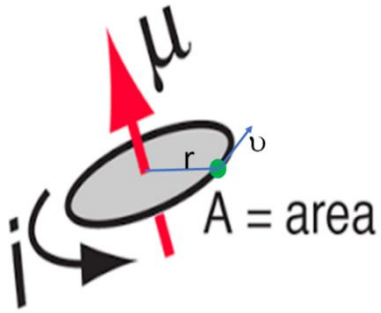


Figure 1. Schematic diagram of a particle moving along the circumference of circle generates a magnetic moment given by Eq. 1.

A particle with mass m charge q revolving at the circumference of a circle of radius r as shown in Fig. 1 has an angular momentum $\vec{L}$ and generates a magnetic moment $\vec{\mu}$ given by Eq. 1 below.

$$\vec{\mu} = IA\hat{n} = \frac{q\upsilon}{2\pi r}\pi r^2\hat{n} = \frac{q}{2m}(m\upsilon r)\hat{n} = \frac{q}{2m}\vec{L} \quad (1)$$

The connection of the magnetic moment generated by a classical particle as describe by Eq. 1 and the magnetic moments of an elementary particle like electron or proton is given by Eq. 2

$$\vec{\mu} = \frac{q}{2m}\vec{L} = \frac{q}{2m}g\vec{S} = \frac{q}{2m}(\frac{g-2}{2}+1)\vec{S} = \frac{q}{m}(G+1)\vec{S} \quad (2)$$

In Eq. 2 the symbol "g" is the gyromagnetic ratio, where G=(g-2)/2 is the anomalous magnetic moment of the particle and $\vec{S}$ is the spin of the elementary particle.

A stationary particle with magnetic moment $\vec{\mu}$ in a magnetic field $\vec{B}$ experiences a torque $\vec{\tau}$ and the magnetic moment precesses about the direction of the magnetic field with an angular frequency $\vec{\Omega}_{prec} = -\frac{q\vec{B}}{m_0}\frac{g}{2}$ as it is shown by the left part of Fig. 2 and described by the Eq. 3 below.

$$\frac{d\vec{S}}{dt} = \vec{\tau} = \vec{\mu}\times\vec{B} = \frac{q}{2m}g\vec{S}\times\vec{B} = -\frac{q}{2m}g\vec{B}\times\vec{S} = \vec{\Omega}_{prec}\times\vec{S} \quad (3)$$

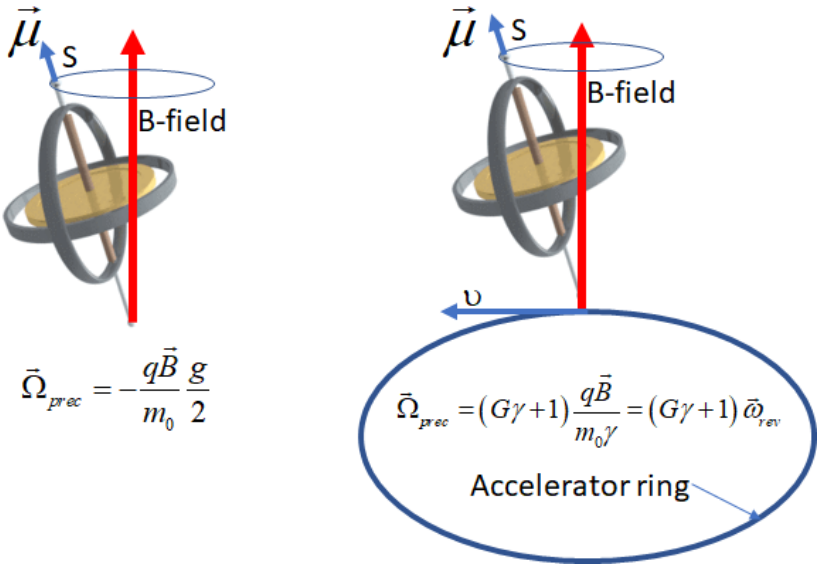


Figure 2. Schematic picture of a spinning particle with magnetic moment $\vec{\mu}$, and spin $\vec{S}$ in static magnetic field $\vec{B}$ (left). Right picture is same as the left but the particle is moving in the guiding field of an accelerator.

When the particle moves along an orbit due to the magnetic field $\vec{B}$ of an accelelator schematically shown by the right picture in Fig. 2, the angular frequency $\vec{\Omega}_{prec}$ of the pecession is given by Eq. 4.

$$\vec{\Omega}_{prec} = (G\gamma+1)\frac{q\vec{B}}{m_0\gamma} = (G\gamma+1)\vec{\omega}_{rev} \quad (4)$$

In Eq. 4 $\vec{\omega}_{rev} = \frac{q\vec{B}}{m_0\gamma}$ is the angular frequency of revolution of a particle in the acceleration.

Eq. 4 can be derived from the BMT [4] equation which describes the spin motion of a particle in an electromagnetic field.

In an accelerator or a beam line with vertical guiding dipole fields along the vertical, the stable spin direction is along the vertical therefore transferring a polarized beam from a beam line to an accelerator or vise versa the stable spin direction of the beam line and the accelerator is always matched because the beam line and the accelerator maintain the vertical stable spin direction. A particle accelerator which is often equipped with magnetic devices like helical magnets [8] which help preserve the polarization of the beam during acceleration the stable spin direction is not always along the vertical and depends on the azimuthal angle of the accelerator as well as the momentum of the beam. Accelerators equipped with helical magnets the matching of the stable spin direction at the injection/extraction point of the accelerator is a problem that must be coped with. The first part of the study in this technical note is the calculation of the stable spin direction in an accelerator and of a beam line using the method of spinors. Another important physical quantity when dealing with polarized beams is the "spin tune" ν_{st} of a particle in an accelerator which is defined as the number of spin precessions about the stable spin direction per

revolution. In the particular case when the stable spin direction of an accelerator is along the vertical the spin tune ν_{st} can be calculated from Eq. 4 above and it is equal to $\nu_{st} = \frac{\Omega_{prec} - \omega_{rev}}{\omega_{rev}} = G\gamma$ [8]. The spin tune ν_{st} which is also part of the study in this technical note depends on the momentum of the particle during the acceleration and is related to the periodic spin motion in an accelerator. If the spin tune of a particle during its acceleration has "particular values" the spin of the particle is subject to resonance conditions. These resonances are called "spin imperfection resonances", and "intrinsic spin resonances". The spin imperfection resonances occur when the condition $G\gamma=n$ is satisfied and the spin intrinsic resonances occur when the condition $G\gamma=n+ Q_{x,y}$ is satisfied. The symbol $Q_{x,y}$ is the horizontal (Q_x) and vertical (Q_y) betatron tunes respectively. These resonances can change the spin direction of the particles in the beam bunch with the result of possible loss of the polarization of the whole beam in the bunch.

General comments on the Stable Spin Direction of a beam line or an accelerator.

When a polarized beam is transferred from a beam line into an accelerator, in addition to the matching of the beam parameters, the "matching of the stable spin direction" is also required at the location of the injection point. Below is a list of conditions that the stable spin direction must satisfied in an accelerator or a beam line.

- The stable spin direction of the reference particle in an accelerator must be periodic at any point along the orbit. This condition uniquely defines the stable spin direction of the reference particle along any point of the closed orbit.
- In a beam line the stable spin direction depends on the strength of the magnetic or electric elements of the line as well as on the stable spin direction of the reference particle at the entrance of the beam line.
- The matching of the stable spin direction of an injected beam into an accelerator can be accomplished by either:
    - changing the stable spin direction of the reference particle of the circulating beam in the accelerator or
    - changing the stable spin direction of the reference particle of the injected beam at the injection point.

  In some cases of spin matching only one of the two possibilities mentioned above can be used for the matching of the stable spin direction.

Fig. 3 is a schematic diagram of the stable spin direction of an injection line, and an accelerator. In the top figure, the stable spin direction of the beam line and of the accelerator is along the vertical. In the bottom figure the stable spin direction of both, the beam line and the accelerator are not along the vertical. "Stable spin direction matching" requires that the stable spin direction of the beam line and of the accelerator should be colinear at the injection point.

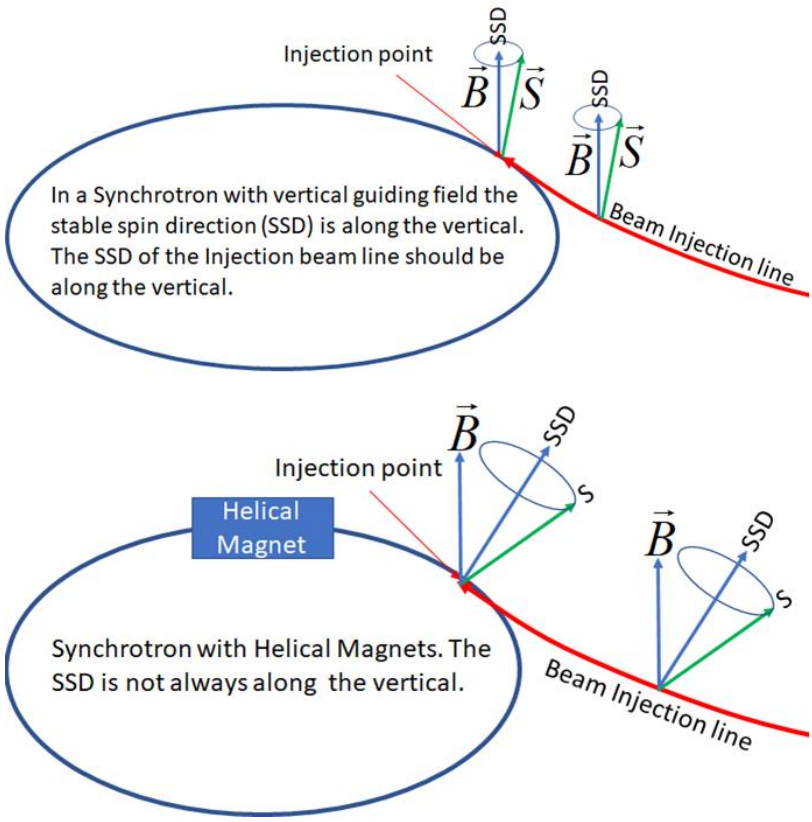


Figure 3. In the top figure, the stable spin direction of the beam line and of the accelerator is along the vertical. In the bottom figure the stable spin direction of both the beam line and the accelerator are not along the vertical. This requires that the stable spin direction of the beam line and the accelerator should be colinear at the injection point. "matching of the stable spin direction".

Methods to calculate the stable spin direction and spin tune ν_{st} in a beam line or a synchrotron.

The calculation of the stable spin direction of the reference particle in an accelerator or a beam line can be done with either method described below.

- Integrate the equation of motion (Lorentz equation) and the spin motion (BMT equation) of the reference particle moving in the E and B fields of the accelerator or beam line: This is the classical description of particle motion and of its spin in an electromagnetic field.
    - Lorentz force $\vec{F} = q(\vec{E} + \vec{v} \times \vec{B})$ Both the E and B fields are functions of six dimensional coordinates (x, y, z. v_x. v_y, v_z, dp/p).
    - BMT equation $\frac{d\vec{S}}{dt} = \vec{\Omega} \times \vec{S}$, $\vec{\Omega} = -\frac{e}{mc}\left[\left(G + \frac{1}{\gamma}\right)\vec{B} - \frac{G\gamma}{\gamma+1}\left(\vec{\beta}\bullet\vec{B}\right)\vec{\beta} - \left(G + \frac{1}{\gamma+1}\right)\vec{\beta} \times \vec{B}\right]$ $\vec{S} = \left(S_x, S_y, S_z\right)$

- Use of Spinors which are the eigenstates that describe the spin state of a particle:
    - As the particle moves through a magnetic field its spin state is rotated (transforms) by each magnetic field element about the stable spin direction of the element.
    - Using the Pauli matrices which are the operators acting on the spin states (spinors) one calculates the rotation matrix **R** as the spin of the particle moves in the magnetic field of each element and therefore the total rotation matrix $\mathbf{R}_{tot} = \mathbf{R}_n \bullet \mathbf{R}_{n-1} \bullet \bullet \bullet \mathbf{R}_2 \bullet \mathbf{R}_1$ can be calculated as the product of the individual rotation matrices. .

The first method is more accurate because it calculates the infinitesimal spin rotation matrix at any given point along the trajectory of the particle. This method of using the integration of the charged particle motion and its spin motion, has been used in refs. [1,2,3]. This technical note describes the spinor-method to calculate the stable spin direction of a beam line or an accelerator. The "spinors" method has been used in Ref. [9,10] and requires the knowledge of the stable spin direction of a particle in each of the magnetic element of the beam line or the accelerator.

When the Spinor method is used this stable spin direction of a particular magnetic element is assumed to be known by prior usage of the integration of the "Lorentz and BMT" equations for each particular element of the line or accelerator.

The spinor method of calculating the Stable Spin Direction.

The rotated function $\psi(x',y',z')$ of a scaler field about an axis $\hat{n}$ is by definition the original scalar function $\psi(x,y,z)$ expressed in the rotated coordinates $\psi(x',y',z') = \mathbf{R(n,\theta)}\psi(x,y,z)$ as shown in Fig. 4.

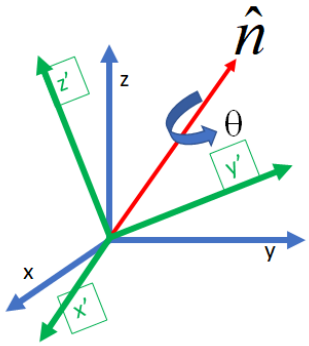


Figure 4. The rotation $\mathbf{R(n,\theta)\psi(x,y,z)}$ of a scalar function $\psi(x,y,z)$ is the original scalar function expressed in the rotated coordinate system (x',y',z') { $\psi(x',y',z')$ }.

The expression of the rotation operator $\mathbf{R}(\hat{n},\theta)$ of the angular momentum operator L of the wave function is derived in Appendix I to be equal $\mathbf{R}(\hat{n},\theta) = e^{-i\theta(\hat{n}\cdot\vec{L})}$ (5)

In Eq. 5 $\hat{n}$ is the axis of rotation, θ the rotation angle and $\vec{L}$ is the angular momentum operator.

The rotation operator [11] for spin ½ is derived in Appendix II.

The subsection below lists the operators for spin ½ particles.

Spin Operators for spin ½ particles

To derive the spin Rotation operators S_x, S_y, S_z for spin ½ particle which are required to calculate the stable spin direction on must derive first the eigenstates for the spin ½ particles (spinors) (see Appendix II) based on the experimental results from the Stern Gerlach experiment [6]. The Rotation operators S_x, S_y, S_z for spin ½ particles are also derived in Appendix II using the eigenstates and the eigenvalues of spin ½ particles. the spin operators S_x, S_y, S_z for spin ½ particle.

The Table I below summarizes the angular momentum operators of spin ½ particles in the form of 2x2 matrices and the eigenstates for spin ½ particles in the form of on column vector. The coordinate system corresponding to the directions referred in the Table I is shown in Fig. 5.

Table I. The spin operators and the eigenstates of spin ½ particles. The coordinate system is shown in Fig. 5.

	Operator	*Eigenvector* #1	*Eigenvector* #2
$X-Dir$	$S_x = \frac{\hbar}{2}\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix}$	$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ -1 \end{pmatrix}$
$Y-Dir$	$S_y = \frac{\hbar}{2}\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$	$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ i \end{pmatrix}$	$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ -i \end{pmatrix}$
$Z-Dir$	$S_z = \frac{\hbar}{2}\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	$\begin{pmatrix} 1 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 1 \end{pmatrix}$

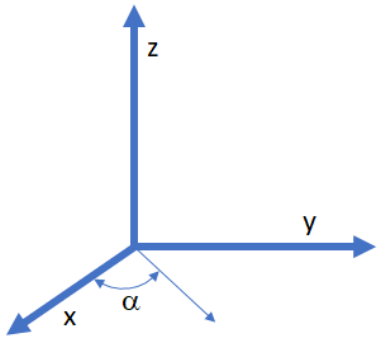


Figure 5. The coordinate system corresponding to the spin ½ matrices and of the eigenstates of spin ½ particles appearing in Table 1.

Table II is the same as Table I but it lists the Pauli matrices instead of the spin ½ operators.

Table II. The Pauli Matrices and the eigenstates of spin ½ particles. The coordinate system is shown in Fig. 5.

	Pauli Matrix	*Eigenvector* #1	*Eigenvector* #2
$X-Dir$	$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix}$	$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ -1 \end{pmatrix}$
$Y-Dir$	$\sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$	$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ i \end{pmatrix}$	$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ -i \end{pmatrix}$
$Z-Dir$	$\sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	$\begin{pmatrix} 1 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 1 \end{pmatrix}$

Rotation matrices for spin ½ particles

In Appendix I the rotation matrix about the i[th] axis (x,y,z) is given by $R_i(\theta) = e^{-i\theta_i L_i}$

Table III list the rotation matrices of the spin ½ particle along the x,y,z directions and the corresponding Pauli matrices for spin ½ particles.

Table III. The rotation matrices of the spin ½ particle along the x,y,z directions and the corresponding Pauli matrices for spin ½ particles. The coordinate system is shown in Fig. 5.

	Rot. matrix	*Pauli matrix*
$X-Dir$	$e^{-i\hbar\frac{\theta_x}{2}\sigma_x}$	$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
$Y-Dir$	$e^{-i\hbar\frac{\theta_y}{2}\sigma_y}$	$\sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$
$Z-Dir$	$e^{-i\hbar\frac{\theta_z}{2}\sigma_z}$	$\sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$

From Appendix III the rotation matrix about the x,y,z axes for spin ½ particle can be also expressed as:

$$R_x(\frac{\theta_x}{2}) = e^{-i\frac{\theta_x}{2}\sigma_x} = I\cos\frac{\theta_x}{2} - i\sigma_x\sin\frac{\theta_x}{2} \quad (6)$$

$$R_y\left(\frac{\theta_y}{2}\right) = e^{-i\frac{\theta_y}{2}\sigma_y} = I\cos\frac{\theta_y}{2} - i\sigma_y\sin\frac{\theta_y}{2} \quad (7)$$

$$R_z\left(\frac{\theta_z}{2}\right) = e^{-i\frac{\theta_z}{2}\sigma_z} = I\cos\frac{\theta_z}{2} - i\sigma_z\sin\frac{\theta_z}{2} \quad (8)$$

If the rotation of the spinor eigenstate in along the $\hat{n}$ direction the projection of the spin ½ operator is

$S_n = \vec{\sigma}\cdot\hat{n}$ and the spin rotation matrix is given Eq. 5. (see Appendix I and II)

$$\mathbf{R}(\hat{n},\theta) = e^{-i(\vec{\sigma}\cdot\hat{n})\frac{\theta}{2}} = \mathbf{I}\cos\frac{\theta}{2} - i(\vec{\sigma}\bullet\hat{n})\sin\frac{\theta}{2} \quad (9)$$

Implementation of the spinor method for calculating the stable spin direction of a beam line and a synchrotron and the spin tune of a synchrotron

In the previous sections we discussed the formulation which leads to the expressions of the spin rotation matrices which are needed to transform the ½ spin particles as they are transported along a beam line or

circulate in an accelerator. Using the cumulative spin rotation matrix $\mathbf{R}_{tot} = \mathbf{R}_n \bullet \mathbf{R}_{n-1} \bullet \bullet \bullet \mathbf{R}_2 \bullet \mathbf{R}_1$ the determination of the stable spin direction at any given point along of a beam line or accelerator and the spin tune in an accelerator is discussed in this section. This section is split in three subsections;

a) the subsection dealing with the stable spin direction along a beam line,
b) the subsections of the stable spin direction along an accelerator, and the
c) subsection dealing with the determination of the spin tune of the reference particle in a synchrotron.

Stable spin direction of a beam line.

This section discusses the stable spin direction of particle with spin ½ transported along a beam line. The concept of spin tune for a beam line is not relevant because a beam line being not a periodic structure or if it is periodic; It a periodic structure with only few periods.

There are a couple of equivalent methods to calculate the stable spin direction of a beam line by the use of spinors.

Method 1:

• From the classical stable spin direction $S_{in}=(S_x, S_y, S_z)_{in}$ at the entrance of the beam line determine the spin wavefunction which a superposition of the basic |+> and |-> spinors.

$$\psi_{entr}(x,y,z) = a_{in}\left|+\right\rangle + b_{in}\left|-\right\rangle = a_{in}\begin{pmatrix}1\\0\end{pmatrix} + b_{in}\begin{pmatrix}0\\1\end{pmatrix} \quad (10)\,.$$

The amplitudes a_{in} and b_{in} are calculated in terms of the spin components of the classical spin direction $S_{in}=(S_x, S_y, S_z)_{in}$ in ref. [10] and in Appendix V and are given by:

$$\Psi_{entr} = \begin{pmatrix} a_{entr} \\ b_{entr} \end{pmatrix} = a_{entr}\begin{pmatrix}1\\0\end{pmatrix} + b_{entr}\begin{pmatrix}0\\1\end{pmatrix} = -i\frac{\left(S_x - iS_y\right)_{ent}}{\sqrt{2\left(1-S_z\right)_{entr}}}\begin{pmatrix}1\\0\end{pmatrix} + i\frac{\left(1-S_z\right)_{entr}}{\sqrt{2\left(1-S_z\right)_{entr}}}\begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix} -i\dfrac{\left(S_x - iS_y\right)_{entr}}{\sqrt{2\left(1-S_z\right)_{entr}}} \\ i\dfrac{\left(1-S_z\right)_{entr}}{\sqrt{2\left(1-S_z\right)_{entr}}} \end{pmatrix} \quad (11)$$

• The calculation of the final 2x2 rotation matrix of the spinor follows by multiplying the individual spin rotation matrices of each magnetic element.

$$\mathbf{R}_{final}\left(\hat{n},\theta\right) = \mathbf{R}_n\left(\hat{n},\theta_n\right)\mathbf{R}_{n-1}\left(\hat{n},\theta_{n-1}\right)....\mathbf{R}_2\left(\hat{n},\theta_2\right)\mathbf{R}_1\left(\hat{n},\theta_1\right) \quad (12)$$

Each individual rotation matrix is expressed as:

$$\mathbf{R}_i\left(\hat{n},\theta_i\right) = I\cos\frac{\theta_i}{2} + i(\vec{\sigma}\bullet\hat{n})\cos\frac{\theta_i}{2} \quad (13)$$

The spinor at the exit of the beam line can be calculated by the relation (9) below.

$$\Psi_{exit} = \mathbf{R}_{final}\left(\hat{n},\theta\right)\Psi_{entr} \quad (14)$$

Now using the expression of the spinors in term of the three spin components (S_x, S_y, S_z) given by the relation (15) below which is derived in Appendix V and in the classical spin components (directional cosines) (S_x, S_y, S_z) of the stable spin direction at the exit of the beam line can be calculated.

$$\Psi_{exit} = \begin{pmatrix} a_{exit} \\ b_{exit} \end{pmatrix} = a_{exit}\begin{pmatrix} 1 \\ 0 \end{pmatrix} + b_{exit}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = i\frac{\left(S_x - iS_y\right)_{exit}}{\sqrt{2\left(1-S_z\right)_{exit}}}\begin{pmatrix} 1 \\ 0 \end{pmatrix} + \frac{i\left(1-S_z\right)_{exit}}{\sqrt{2\left(1-S_z\right)_{exit}}}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = i\begin{pmatrix} \frac{\left(S_x - iS_y\right)_{exit}}{\sqrt{2\left(1-S_z\right)_{exit}}} \\ \frac{\left(1-S_z\right)_{exit}}{\sqrt{2\left(1-S_z\right)_{exit}}} \end{pmatrix} \quad (15)$$

Method 2:

An equivalent way of computing the directional cosines of the stable spin direction at the exit of the beam line is given by the relation (16) below.

$$\vec{S}_{exit} = (S_x, S_y, S_z)_{exit} = \Psi^*_{exit}\hat{\sigma}\Psi_{exit} = \Psi^*_{exit}\begin{pmatrix}\sigma_x & \sigma_y & \sigma_z\end{pmatrix}\Psi_{exit} =$$
$$\left(\Psi^*_{entr}\mathbf{R}^T_{final}\right)\begin{pmatrix}\sigma_x & \sigma_y & \sigma_z\end{pmatrix}\left(\mathbf{R}_{final}\Psi_{entr}\right) = \Psi^*_{entr}\left(\mathbf{R}^T_{final}\right)\begin{pmatrix}\sigma_x & \sigma_y & \sigma_z\end{pmatrix}\left(\mathbf{R}_{final}\right)\Psi_{entr} \quad (16)$$

Method 3:

Another equivalent way is to calculate directional cosines of the stable spin direction appears in Ref. [12]

In this rererance the 3x3 rotation matrix is calculated from the elements of the 2x2 rotation matrix.

The expression of the matrix elements of R_{3x3} rotation matrix in term of the matrix elements of the R_{2x2} rotation matrix is given below.

$$\mathbf{R}_{3x3} = \begin{pmatrix} \mathrm{Re}(R_{11}R_{11} - R_{12}R_{12}) & \mathrm{Im}(R_{11}R_{11} + R_{12}R_{12}) & -2\,\mathrm{Re}(R_{11}R_{12}) \\ -\mathrm{Im}(R_{11}R_{11} - R_{12}R_{12}) & \mathrm{Re}(R_{11}R_{11} + R_{12}R_{12}) & 2\,\mathrm{Im}(R_{11}R_{12}) \\ 2\,\mathrm{Re}(R_{11}\bar{R}_{12}) & 2\,\mathrm{Im}(R_{11}\bar{R}_{12}) & \left|R_{11}\right|^2 - \left|R_{12}\right|^2 \end{pmatrix} \quad (17)$$

Using the 3x3 rotation matrix given by the relation (13) the directional cosines of the stable spin direction at the exit are given by:

$$\vec{S}_{exit} = \begin{pmatrix} S_x \\ S_y \\ S_z \end{pmatrix}_{exit} = \mathbf{R}_{3x3}\begin{pmatrix} S_x \\ S_y \\ S_z \end{pmatrix}_{entr} \quad (18)$$

Stable spin direction of the reference orbit in an accelerator.

The spin of the reference particle circulating along the closed orbit of an accelerator with dipole guiding field the stable spin direction is along the vertical. To preserve the polarization of the circulating beam during the acceleration, the accelerator employs helical magnets which change the stable spin direction from the vertical. These helical magnets cause the stable spin direction to vary along the closed reference orbit.

The rest of this section describes a method to calculate the stable spin direction at a particular angle along the reference orbit, using the spinor's method. This method was used in Ref. [9,10].

The problem is to calculate the stable spin direction at a particular point along the reference orbit of the accelerator.

It is assumed that the accelerator consists of dipole guiding magnets with vertical fields and helical magnets in between the dipole magnets.

Fig. 6 helps in the explanation of the determination of the stable spin direction at a particular point along the reference orbit. It is a schematic diagram of a synchrotron which employs two helical magnets.

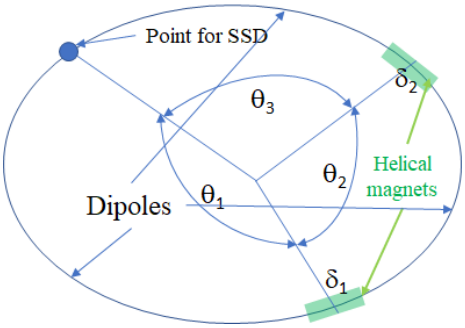


Figure 6. A schematic picture showing a synchrotron with dipole magnets along the arc-span θ_1, followed by a helical magnet which rotates the spin by an angle δ_1 about the $\hat{n}_1$ axis, followed by dipole magnets which bend the beam by an angle θ_2, helical magnet which rotates the spin by an angle δ_2 about the $\hat{n}_2$ axis, and dipole magnets which bend the beam by an angle θ_3 to closed the orbit.

To calculate the stable spin direction at a particular point indicating in Fig. 6 one has to construct the rotation matrix along the reference orbit according to the relation (14) below.

$$\mathbf{R}_{synch} = \mathbf{R}_{dip}(\theta_3)\mathbf{R}_{hm}(\delta_2)\mathbf{R}_{dip}(\theta_2)\mathbf{R}_{hm}(\delta_1)\mathbf{R}_{dip}(\theta_1) \qquad (14)$$

$$\mathbf{R}_{dip}(\theta) = \cos\frac{\theta}{2} - i\sigma_z \sin\frac{\theta}{2} \qquad (15)$$

$$\mathbf{R}_{hm}(\theta) = \cos\frac{\delta}{2} - i(\vec{\sigma}\bullet\hat{n})\sin\frac{\delta}{2} \qquad (16)$$

Substituting the expressions of the individual rotation matrices of the magnetic elements from Eq. (15) and (16) into the Eq. (14) the rotation matrix $\mathbf{R}_{synchr}$ is obtained in terms of the individual rotation matrices of the elements which comprise the synchrotron.

The spin rotation angle about the stable spin direction after one revolution in the synchrotron is given by [12]

$$\cos\frac{\theta}{2} = Tr\left(\mathbf{R}_{synch}\right) \quad (17)$$

Note that for a dipole magnet the angle of precession $\theta = G\gamma\theta_{bend}$. This equation is easily derived from the general BMT equation, and θ_{bend} is angle of bend of the charged particle in the dipole.

Eq. (17) can be proved by considering that the general form of the rotation matrix for spin ½ particles is given by Eq. AI-7 in Appendix I which can be written as:

$$\mathbf{R}(\hat{n},\theta) = e^{-i\theta(\vec{\sigma}\bullet\hat{n})} = \mathbf{I}\cos\frac{\theta}{2} - i\left(\left(\vec{\sigma}\bullet\hat{n}\right)\sin\frac{\theta}{2}\right. \quad (18)$$

Thus

$$Tr\left(\mathbf{R}(\hat{n},\theta)\right) = Tr\left(\mathbf{I}\cos\frac{\theta}{2} - i\left(\vec{\sigma}\bullet\hat{n}\right)\sin\frac{\theta}{2}\right) = \mathrm{Tr}\left(\mathbf{I}\cos\frac{\theta}{2}\right) - i\sin\frac{\theta}{2}\mathrm{Tr}\left(\vec{\sigma}\bullet\hat{n}\right) = 2\cos\frac{\theta}{2} \quad (19)$$

The trace of: $\mathrm{Tr}\left(\vec{\sigma}\bullet\hat{n}\right) = Tr\left(\hat{n}_x\sigma_x + \hat{n}_y\sigma_y + \hat{n}_z\sigma_z\right) = \mathrm{Tr}\begin{pmatrix} \hat{n}_z & \hat{n}_x - i\hat{n}_y \\ \hat{n}_x + i\hat{n}_y & -\hat{n}_z \end{pmatrix} = 0 \quad (20)$

$$Tr\left(\vec{\sigma}\mathbf{R}(\hat{n},\theta)\right) = Tr\left(\vec{\sigma}\mathbf{I}\cos\frac{\theta}{2} - i\vec{\sigma}\left(\vec{\sigma}\bullet\hat{n}\right)\sin\frac{\theta}{2}\right) = \mathrm{Tr}\left(\vec{\sigma}\mathbf{I}\cos\frac{\theta}{2}\right) - i\sin\frac{\theta}{2}\mathrm{Tr}\left(\vec{\sigma}\left(\vec{\sigma}\bullet\hat{n}\right)\right)$$

$$\cos\frac{\theta}{2}Tr\left(\vec{\sigma}\mathbf{I}\right) - i\sin\frac{\theta}{2}\mathrm{Tr}\left(\vec{\sigma}\left(\vec{\sigma}\bullet\hat{n}\right)\right) \quad (21)$$

In Eq. (21) $Tr\left(\vec{\sigma}\mathbf{I}\right) = 0$ (22) and $\left(\vec{\sigma}\left(\vec{\sigma}\bullet\hat{n}\right)\right) = \hat{n}\mathbf{I} + i\left(\hat{n}\times\vec{\sigma}\right)$ (23)

Thus Eq. 21 becomes:

In Eq. (24) $\mathrm{Tr}\left(\hat{n}\times\vec{\sigma}\right) = 0$ (24)

Thus: $\hat{n} = \left[\frac{i}{2\sin\frac{\theta}{2}}\right] Tr\left(\vec{\sigma}\mathbf{R}(\hat{n},\theta)\right)$ (25)

Spin tune of the reference particle in an accelerator.

The spin tune of the reference particle in a synchrotron is the number of spin precessions per turn.

If the total angle of the spin precession per turn is θ the spin tune is $\nu_{st} = \frac{\theta}{2\pi}$. Substituting the angle θ in Eq. 17 the spin tune is given by:

$$\cos\frac{2\pi\nu_{st}}{2} = \cos\left(\pi\nu_{st}\right) = Tr\left(\mathbf{R}_{synch}\right) \qquad (26)$$

Computer codes.

A computer code which calculates the stable spin direction at the exit of the AtR line [10] using the spinor method, is included in Appendix VI.

References

[1] N. Tsoupas et. al "Transfer of a Polarized Proton Beam from AGS to RHIC" PAC97 Vancouver, BC May 12-16 1997.

[2] N. Tsoupas et. al "Stable spin Direction od Polarized Proton Beam at the Injection Point of RHIC", BNL-103647-2014-TECH AGS/RHIC/SN 021

[3] N. Tsoupas, and T. Roser, "A proposed simple Modification of the AtR Line to Optimize the Polarization Transfer of a Polarized Proton Beam from AGS to RHIC" AGS/RHIC/SN No. 022

[4] Bergmann V, Michel L and Telegdi V L 1959 Phys. Rev. Lett. 2 435–6

[5] S. Goudsmit and G.E. Uhlenbeck, Physica **6** (1926) 273.

[6] After its conception by Otto Stern in 1921, the experiment was first successfully conducted by Walther Gerlach in early 1922.[1][4][5]

[7] In 1928 in England Paul Dirac found an equation that was fully compatible with Special Relativity

[8] BMT Equation Analysis and Spin in Accelerator Physics T. Zolkin† The University of Chicago, Department of Physics

[9] N. Tsoupas et al. https://www.researchgate.net/publication/255001822_The_AGS_synchrotron_with_four_helical_magnets

[10] N. Tsoupas et. al. **Polarized $^{3}He^{+2}$ ions in the Alternate Gradient Synchrotron to RHIC transfer line**

link.aps.org › pdf › PhysRevAccelBeams.19.094702

[11] David H. McIntyre “Spin and Quantum Measurements” Oregon State University
[12] D. B. Westra SU(2) and SO(3) SU(2) and SO(3)www.mat.univie.ac.at › ~westra › so3su2
[13] B. W. Montague 1-s2.0-0370157384900310-main.pdf
[14] S. B. Kowalski, H. A. Enge https://doi.org/10.1016/0168-9002(87)90921-1

Appendix I

Rotation Operator of an angular momentum operator L.

The rotation operator R of a spin state transforms the original spin state ψ(x,y,z) to a new state ψ(x′,y′,z′) which is expressed by AI-1.

$$\psi(x',y',z') = R\,\psi(x,y,z) \quad \text{(AI-1)}$$

To derive an expression of the rotation operator we consider an infinitesimal rotation about the z-axis by an angle dθ. The coordinates of the original coordinate system are related to those of the rotated coordinate system by the relations AI-2 below.

$$x' = x + yd\theta \qquad y' = y - xd\theta \quad \text{(AI-2)}$$

Expanding in McLauren series the rotate function ψ(x′,y′,z′)

$$\psi(x',y',z') = \psi(x',y',z) = \psi(x,y,z) + \frac{\partial\psi}{\partial x}dx + \frac{\partial\psi}{\partial y}dy = \psi(x,y,z) + \frac{\partial\psi}{\partial x}yd\theta - \frac{\partial\psi}{\partial y}xd\theta$$

$$\psi(x',y',z') = \psi(x,y,z) + d\theta\left(y\frac{\partial}{\partial x} - x\frac{\partial}{\partial y}\right)\psi(x,y,z) = (1 - id\theta L_z)\psi(x,y,z) \quad \text{(AI-4)}$$

The L_z in Eq. AI-4 is $L_z = -i\left(y\frac{\partial}{\partial x} - x\frac{\partial}{\partial y}\right)$ (AI-5)

Similar expressions are derived for the L_x, and L_y.

$$L_x = -i\left(y\frac{\partial}{\partial z} - z\frac{\partial}{\partial y}\right) \qquad L_y = -i\left(z\frac{\partial}{\partial x} - x\frac{\partial}{\partial z}\right) \qquad L_z = -i\left(y\frac{\partial}{\partial x} - x\frac{\partial}{\partial y}\right) \quad \text{(AI-5)}$$

The expressions in (AI-5) are the angular momentum operators along the x,y,z directions.

A finite rotation about the z-axis is the sum of many rotations as is shown in the relation (AI-6) below.

$$R_z = \lim_{n\to\infty}\left(1 - id\theta L_z\right)^n = \left(1 - i\frac{\theta_z}{n}L_z\right)^n = e^{-i\theta_z L_z}$$

$$\ln R_z = \ln\lim_{n\to\infty}\left(1 - id\theta L_z\right)^n = \ln\lim_{n\to\infty}\left(1 - i\frac{\theta_z}{n}L_z\right)^n = \lim_{n\to\infty}\ln\left(1 - i\frac{\theta_z}{n}L_z\right)^n = \lim_{n\to\infty} n\ln\left(1 - i\frac{\theta_z}{n}L_z\right)$$

$$\ln R_z = \lim_{n\to\infty} n\ln\left(1 - i\frac{\theta_z}{n}L_z\right) = \lim_{n\to\infty} n\left(-i\frac{\theta_z}{n}L_z - \frac{1}{2}\left(-i\frac{\theta_z}{n}L\right)^2 + \frac{1}{3}\left(-i\frac{\theta_z}{n}L\right)^3 - \ldots\ldots\right) = -i\theta_z L_z$$

$$R_z(\theta) = e^{-i\theta_z L_z} \quad \text{(AI-6)}$$

If the Rotation is about an axis with unit vector $\hat{n}$, the totation operator is given by Eq. AI-7 below.

$$\mathbf{R}(\hat{n},\theta) = e^{-i\theta\left(\vec{S}\bullet\hat{n}\right)} \quad \text{(AI-7)}$$

In the relation above the quantity $\vec{S}\bullet\hat{n}$ is the projection of the spin operator along the $\hat{n}$ direction.

In the next Appendix II the expression of the of the spin rotation operation for spin ½ particles is derived.

Appendix II

Derivation of Spin ½ particle Spin-Eigenstates (Spinors) and their Operators (Pauli matrices).

This Appendix shows the derivation of the spin ½ particle eigenstates and their operators [11] along the x,y,z directions.

The results of the Stern Gerlach experiment [6] were explained by the hypothesis of the existence of electron spin and also that the electron spin is quantized. This experiment was the basis of the theory which explains the observations of the Stern Gerlach experiment.

Ref. [11] is a detailed description of this theory which is summarized in this Appendix.

The derivation of the $S_{(1/2,z)}$ operator for spin ½ particles

The two spin eigenstates of the electron or of a any particle with ½ spin along the z-direction are defined by AII_1.

$$\psi_z(\pm)=\begin{cases} |+\rangle_z=\begin{pmatrix}1\\0\end{pmatrix} \\ |-\rangle_z=\begin{pmatrix}0\\1\end{pmatrix} \end{cases} \quad \text{(AII\_1)}$$

The $S_{(1/2,z)}$ matrix which the spin operator for the above spin eigenstates can be obtained by solving the two equations below.

$$\left.\begin{aligned} \begin{pmatrix}a & b\\ c & d\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix}&=+\frac{\hbar}{2}\begin{pmatrix}1\\0\end{pmatrix} \\ \begin{pmatrix}a & b\\ c & d\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix}&=-\frac{\hbar}{2}\begin{pmatrix}0\\1\end{pmatrix} \end{aligned}\right\} \Rightarrow \quad S_{(1/2,z)}=\frac{\hbar}{2}\begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix} \quad \text{(AII\_2)}$$

The $S_{(1/2,x)}$ operator along x-direction

To derive the $S_{(1/2,x)}$ rotation matrix corresponding to x- direction eigenstates we consider the linear superposition of the $|+\rangle_x$ and $|-\rangle_x$ eigenstates along the x-direction.

$$|+\rangle_x = a|+\rangle_z + b|-\rangle_z \quad \text{and} \quad |-\rangle_x = c|+\rangle_z + d|-\rangle_z \quad \text{(AII\_3)}$$

The Stern Gerlach experiment shows that if we project the $|+\rangle_x$ onto $|+\rangle_z$ the value of ½ is obtained.

The projection of the $|+\rangle_x$ and $|-\rangle_x$ states along the z-direction yields the set of equations AII_4 below.

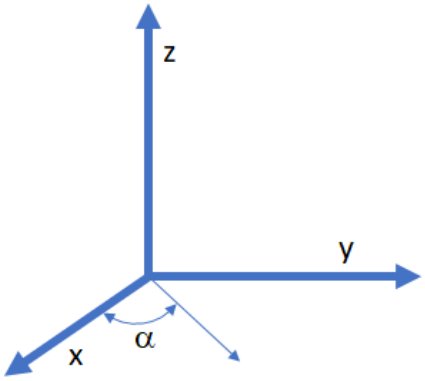
$$\left\{\begin{array}{l} \left|{}_x\langle +||+\rangle\right|^2 = \left|\left(a^*\langle +|+b^*\langle -|\right)|+\rangle\right|^2 = |a^*| = \frac{1}{2} \\ \left|{}_x\langle +||-\rangle\right|^2 = \left|\left(a^*\langle +|+b^*\langle -|\right)|-\rangle\right|^2 = |b^*| = \frac{1}{2} \\ \left|{}_x\langle -||+\rangle\right|^2 = \left|\left(c^*\langle +|+d^*\langle -|\right)|+\rangle\right|^2 = |c^*| = \frac{1}{2} \\ \left|{}_x\langle -||-\rangle\right|^2 = \left|\left(c^*\langle +|+d^*\langle -|\right)|-\rangle\right|^2 = |d^*| = \frac{1}{2} \end{array}\right\} \quad \text{(AII\_4)}$$

Substituting the coefficients from Eqs. AII_4 into AII_3 the spin eigenstates along the x-direction become:

$$\begin{array}{l} |+\rangle_x = \frac{1}{\sqrt{2}}\left(|+\rangle + e^{i\alpha}|-\rangle\right) \\ |-\rangle_x = \frac{1}{\sqrt{2}}\left(|+\rangle + e^{i\beta}|-\rangle\right) \end{array} \quad \text{(AII\_5)}$$

The orthogonality condition in AII_5 ${}_x\langle -||+\rangle_x = 0$ yields $e^{i\alpha} = -e^{i\beta}$.

The choice of phase $\alpha=0^o$ is the correct value for the phase α, as shown in Fig. AII_1, for the eigenstate to be along +x axis. The spin eigenstates along the x-direction become:



$$\begin{array}{l} |+\rangle_x = \frac{1}{\sqrt{2}}\left(|+\rangle + |-\rangle\right) = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix} \\ |-\rangle_x = \frac{1}{\sqrt{2}}\left(|+\rangle - |-\rangle\right) = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\end{pmatrix} \end{array} \quad \text{(AII\_6)}$$

Figure AII_1. The coordinate system.

To calculate the spin operator along the x-axis ($S_{(1/2,x)}$ rotation matrix) we apply the operator on the eigenvector to obtain the $\frac{\hbar}{2}$ eigenvalue.

$$\begin{pmatrix}a & b\\ c & d\end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix} = \frac{\hbar}{2}\frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix} \qquad \begin{pmatrix}a & b\\ c & d\end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\end{pmatrix} = -\frac{\hbar}{2}\frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\end{pmatrix}$$

$$S_{(1/2,x)} = \frac{\hbar}{2}\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad \text{(AII\_7)}$$

The $S_{(1/2,y)}$ operator along y-direction

Similar process as in the derivation of the $S_{(1/2,x)}$ rotation matrix can be followed to derive the $S_{(1/2,y)}$ rotation matrix.

The spin eigenstates can be expressed as a linear combination of the spin eigenstates along the z-direction.

$$\begin{aligned} |+\rangle_y &= a|+\rangle_z + b|-\rangle_z \\ |-\rangle_y &= c|+\rangle_z + d|-\rangle_z \end{aligned} \quad \text{(AII\_8)}$$

By projecting these eigenstates in Eq. AII_8 along the z-direction as it was done for the derivation of the σ_x Pauli matrix we derive the spin eigenstates along the y-direction.

$$\begin{aligned} |+\rangle_y &= \frac{1}{\sqrt{2}}\left(|+\rangle + i|-\rangle\right) = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ i \end{pmatrix} \\ |-\rangle_y &= \frac{1}{\sqrt{2}}\left(|+\rangle - i|-\rangle\right) = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ -i \end{pmatrix} \end{aligned} \quad \text{(AII\_9)}$$

The phase angle α (see Fig. AII_1) in this case is equal to 90°. This is the reason for the appearance of the imaginary symbol in the expression of the eigenstates AII_9.

The matrix elements $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ of the σ_y Pauli matrix operator are calculated by applying this operator on the eigenstates AII_9. The result for the σ_y Pauli matrix operator is given below.

$$S_{(1/2,y)} = \frac{\hbar}{2}\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad \text{(AII\_10)}$$

Table AII_1 summarizes the spin ½ Rotation matrices and the corresponding projection x,y,z eigenstates in the coordinate system shown in Fig AII_1. The coordinate system is shown in Fig. 5.

	Operator	Eigenvalue #1	Eigenvalue #2
X-Dir.	$S_{(1/2,x)} = \frac{\hbar}{2}\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix}$	$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ -1 \end{pmatrix}$
Y-Dir.	$S_{(1/2,y)} = \frac{\hbar}{2}\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$	$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ i \end{pmatrix}$	$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ -i \end{pmatrix}$
Z-Dir.	$S_{(1/2,z)} = \frac{\hbar}{2}\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	$\begin{pmatrix} 1 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 1 \end{pmatrix}$

Table AII_2 below is the same as Table AII_1 above but the operators have been replaced by the Pauli matrices.

Table AII_2 summarizes the spin ½ Pauli rotation matrices and the corresponding projection x,y,z eigenstates in the coordinate system shown in Fig AII_1. The coordinate system is shown in Fig. 5.

	Pauli matrix	Eigenvector #1	Eigenvector #2
X-Dir	$\sigma_{(1/2,x)}=\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}$	$\frac{1}{\sqrt{2}}\begin{pmatrix}1\\ 1\end{pmatrix}$	$\frac{1}{\sqrt{2}}\begin{pmatrix}1\\ -1\end{pmatrix}$
Y-Dir	$\sigma_{(1/2,y)}=\begin{pmatrix}0 & -i\\ i & 0\end{pmatrix}$	$\frac{1}{\sqrt{2}}\begin{pmatrix}1\\ i\end{pmatrix}$	$\frac{1}{\sqrt{2}}\begin{pmatrix}1\\ -i\end{pmatrix}$
Z-Dir	$\sigma_{(1/2,z)}=\begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}$	$\begin{pmatrix}1\\ 0\end{pmatrix}$	$\begin{pmatrix}0\\ 1\end{pmatrix}$

Rotation Operators of spin ½ particle about an axis along the unit vectors $\hat{x},\hat{y},\hat{z}$

In Appendix I the angular momentum rotation operation about the z-axis was dirived to be

$$R_z(\theta)=e^{-i\theta_z L_z} \quad \text{AII\_11}$$

This subsection deals with the derivation of expressions of the spin ½ operators:

Rotation Operators of spin ½ along the x-direction

Substituting in AII_11 the angular momentum operator along the x-direction from Table AII_2

$$R_x(\theta)=e^{-i\theta_x L_x}=e^{-i\theta_x \sigma_x}=e^{-i\frac{\theta_x}{2}\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}} \quad \text{(AII\_12)}$$

Using the expansion:

$$e^{-iz}=1+\frac{(-iz)^1}{1!}+\frac{(-iz)^2}{2!}+\frac{(-iz)^3}{3!}+\frac{(-iz)^4}{4!}+\frac{(-iz)^5}{5!}+\frac{(-iz)^6}{6!}+\frac{(-iz)^7}{7!}.....=$$

$$(1-\frac{z^2}{2!}+\frac{z^4}{4!}-\frac{z^6}{6!}+\frac{z^8}{8!}-....)-i(\frac{z^1}{1!}-\frac{z^3}{3!}+\frac{z^5}{5!}-\frac{z^7}{7!}+....)=\cos z-i\sin z \quad \text{(AII\_13)}$$

And the relations:

$$\sigma_x\sigma_x=\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}=\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}=\mathbf{I} \quad \text{(AII\_14)}$$

The ½ spin rotation matrix in Eq. AII_12 becomes.

$$R_x(\theta_z) = e^{-i\frac{\theta_x}{2}\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}} = \cos\frac{\theta_x}{2} - i\sigma_x \sin\frac{\theta_x}{2} \qquad \text{(AII\_15)}$$

Similarly:

$$R_y(\theta_y) = e^{-i\frac{\theta_y}{2}\begin{pmatrix}0 & -i\\ i & 0\end{pmatrix}} = \cos\frac{\theta_y}{2} - i\sigma_y \sin\frac{\theta_y}{2} \qquad \text{(AII\_16)}$$

$$R_z(\theta_z) = e^{-i\frac{\theta_z}{2}\begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}} = \cos\frac{\theta_z}{2} - i\sigma_z \sin\frac{\theta_z}{2} \qquad \text{(AII\_17)}$$

Appendix III

Rotation Operator of spin ½ particle about an axis along the unit vector $\hat{n}$

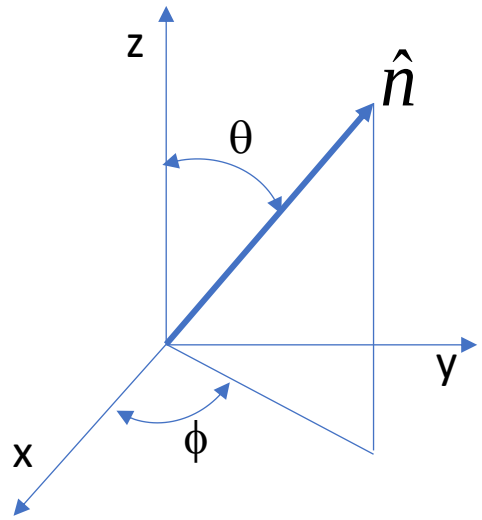


Figure A_III_1. A coordinate system with a unit vector along the $\hat{n}$ direction.

The spin projection along the $\hat{n}$ direction shown in Fig.III_1, is given by the expression A_III_1.

$$S_n = \vec{S}\bullet\hat{n} = S_x \sin\theta\cos\phi + S_y \sin\theta\sin\phi + S_z \cos\phi \qquad \text{(A\_III\_1)}$$

Substituting the expressions of the S_x, S_y, S_z, derived in Appendix II the projection S_n of the spin becomes.

$$S_n = \frac{\hbar}{2}\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\sin\theta\cos\phi + \frac{\hbar}{2}\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}\sin\theta\sin\phi + \frac{\hbar}{2}\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\cos\theta = \frac{\hbar}{2}\begin{pmatrix} \cos\theta & \sin\theta e^{-i\phi} \\ \sin\theta e^{i\phi} & -\cos\theta \end{pmatrix} \qquad \text{(A\_III\_2)}$$

Eigenvalues and eigenvectors of S_n operator

$$S_n = \frac{\hbar}{2}\begin{pmatrix} cos\theta & \sin\theta e^{-i\phi} \\ \sin\theta e^{i\phi} & -\cos\theta \end{pmatrix}$$

$$S_n\begin{pmatrix} a \\ b \end{pmatrix} = \lambda\begin{pmatrix} a \\ b \end{pmatrix}$$

$$\frac{\hbar}{2}\begin{pmatrix} \cos\theta & e^{-i\phi}\sin\theta \\ e^{i\phi}\sin\theta & -\cos\theta \end{pmatrix}\begin{pmatrix} a \\ b \end{pmatrix} = \lambda\begin{pmatrix} a \\ b \end{pmatrix}$$

$$\left(\frac{\hbar}{2}\cos\theta - \lambda\right)a + \frac{\hbar}{2}\left(e^{-i\phi}\sin\theta\right)b = 0$$

$$\frac{\hbar}{2}\left(e^{i\phi}\sin\theta\right)a + \left(-\frac{\hbar}{2}\cos\theta - \lambda\right)b = 0$$

$$\begin{vmatrix} \left(\frac{\hbar}{2}\cos\theta - \lambda\right) & \frac{\hbar}{2}\left(e^{-i\phi}\sin\theta\right) \\ \frac{\hbar}{2}\left(e^{i\phi}\sin\theta\right) & -\left(\frac{\hbar}{2}\cos\theta + \lambda\right) \end{vmatrix} = 0$$

$$+\lambda^2 - \left(\frac{\hbar}{2}\right)^2\cos^2\theta - \left(\frac{\hbar}{2}\right)^2\sin^2\theta = 0 \quad \rightarrow \lambda = \pm\frac{\hbar}{2}$$

Eigenvectors of S_n operator

For $\lambda = +\hbar/2$

$$\left(\cos\theta\right)a + \left(e^{-i\phi}\sin\theta\right)b = a$$

$$|a|^2 + |b|^2 = 1 \qquad |a|^2 + \left|\frac{\cos\theta - 1}{\sin\theta}\right|^2 |a|^2 = 1$$

$$|a|^2 + \left|\frac{\cos^2\frac{\theta}{2} - \sin^2\frac{\theta}{2} - 1}{2\sin\frac{\theta}{2}\cos\frac{\theta}{2}}\right|^2 |a|^2 = 1$$

$$|a|^2+\left|\frac{\sin\frac{\theta}{2}}{\cos\frac{\theta}{2}}\right|^2|a|^2=1$$

$$|a|^2\left(1+\left|\frac{\sin\frac{\theta}{2}}{\cos\frac{\theta}{2}}\right|^2\right)=1 \qquad |a|^2\left(\frac{\sin^2\frac{\theta}{2}+\cos^2\frac{\theta}{2}}{\cos^2\frac{\theta}{2}}\right)=1$$

$$|b|^2=1-\cos^2\frac{\theta}{2}=\sin^2\frac{\theta}{2} \qquad a=\pm\cos\frac{\theta}{2}e^{i\varphi_1} \qquad b=\pm\sin\frac{\theta}{2}e^{i\varphi_2}$$

For $\lambda=-\hbar/2$

$$a=\pm\sin\frac{\theta}{2}e^{i\varphi_1} \qquad b=\pm\cos\frac{\theta}{2}e^{i\varphi_2}$$

$$|+\rangle_n=\begin{pmatrix}a\\b\end{pmatrix}=\begin{pmatrix}\cos\frac{\theta}{2}\\e^{i\phi}\sin\frac{\theta}{2}\end{pmatrix}=\cos\frac{\theta}{2}\begin{pmatrix}1\\0\end{pmatrix}+\sin\frac{\theta}{2}e^{i\phi}\begin{pmatrix}0\\1\end{pmatrix}=\cos\frac{\theta}{2}|+\rangle+\sin\frac{\theta}{2}e^{i\phi}|-\rangle$$

$$|-\rangle_n=\begin{pmatrix}a\\b\end{pmatrix}=\begin{pmatrix}\sin\frac{\theta}{2}\\-e^{-i\phi}\cos\frac{\theta}{2}\end{pmatrix}=\sin\frac{\theta}{2}\begin{pmatrix}1\\0\end{pmatrix}-e^{-i\phi}\cos\frac{\theta}{2}\begin{pmatrix}0\\1\end{pmatrix}=\sin\frac{\theta}{2}|+\rangle-e^{-i\phi}\cos\frac{\theta}{2}|-\rangle$$

$$|+\rangle_n=\begin{pmatrix}a\\b\end{pmatrix}=\begin{pmatrix}\cos\frac{\theta}{2}\\e^{i\phi}\sin\frac{\theta}{2}\end{pmatrix}=\cos\frac{\theta}{2}\begin{pmatrix}1\\0\end{pmatrix}+\sin\frac{\theta}{2}e^{i\phi}\begin{pmatrix}0\\1\end{pmatrix}=\cos\frac{\theta}{2}|+\rangle+\sin\frac{\theta}{2}e^{i\phi}|-\rangle$$

$$|-\rangle_n=\begin{pmatrix}c\\d\end{pmatrix}=\begin{pmatrix}e^{-i\phi}\sin\frac{\theta}{2}\\-\cos\frac{\theta}{2}\end{pmatrix}=e^{-i\phi}\sin\frac{\theta}{2}\begin{pmatrix}1\\0\end{pmatrix}-\cos\frac{\theta}{2}\begin{pmatrix}0\\1\end{pmatrix}=e^{-i\phi}\sin\frac{\theta}{2}|+\rangle-\cos\frac{\theta}{2}|-\rangle$$

$$\hat{A}=\begin{pmatrix}a&b\\c&d\end{pmatrix}|+\rangle=\begin{pmatrix}a&b\\c&d\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix}=\begin{pmatrix}a\\c\end{pmatrix}$$

$$\hat{A}=\begin{pmatrix}\langle+|\hat{A}|+\rangle_n&\langle+|\hat{A}|-\rangle_n\\\langle-|\hat{A}|+\rangle_n&\langle-|\hat{A}|-\rangle_n\end{pmatrix}$$

$$\langle +|\hat{A}|+\rangle_n = \begin{pmatrix}1 & 0\end{pmatrix}\frac{\hbar}{2}\begin{pmatrix}\cos\frac{\theta}{2}\\ e^{i\phi}\sin\frac{\theta}{2}\end{pmatrix} = \cos\frac{\theta}{2} \qquad \langle +|\hat{A}|-\rangle_n = \begin{pmatrix}1 & 0\end{pmatrix}\frac{\hbar}{2}\begin{pmatrix}e^{-i\phi}\sin\frac{\theta}{2}\\ -\cos\frac{\theta}{2}\end{pmatrix} = e^{-i\phi}\sin\frac{\theta}{2}$$

$$\langle -|\hat{A}|+\rangle_n = \begin{pmatrix}0 & 1\end{pmatrix}\frac{\hbar}{2}\begin{pmatrix}\cos\frac{\theta}{2}\\ e^{i\phi}\sin\frac{\theta}{2}\end{pmatrix} = e^{i\phi}\sin\frac{\theta}{2} \qquad \langle -|\hat{A}|-\rangle_n = \begin{pmatrix}0 & 1\end{pmatrix}\frac{\hbar}{2}\begin{pmatrix}e^{-i\phi}\sin\frac{\theta}{2}\\ -\cos\frac{\theta}{2}\end{pmatrix} = -\cos\frac{\theta}{2}$$

$$\hat{S}_n = \left(\vec{S}\bullet\hat{n}\right) = \frac{\hbar}{2}\begin{pmatrix}\cos\frac{\theta}{2} & e^{-i\phi}\sin\frac{\theta}{2}\\ e^{i\phi}\sin\frac{\theta}{2} & -\cos\frac{\theta}{2}\end{pmatrix}$$

$$\mathbf{R}(\hat{n},\theta) = e^{-i\theta\left(\vec{S}\bullet\hat{n}\right)} \quad \text{(AI-7)}$$

APPENDIX V

This Appendix derives the equations (10,11,15) in the main text. It is the relation of the alitudes "a" and "b" of the spinor wavefunction (AV_1 below) and the classical directional cosines (S_x, S_y, S_z) of the ½ spin particles.

$$\Psi = \begin{pmatrix} a \\ b \end{pmatrix} = a\begin{pmatrix} 1 \\ 0 \end{pmatrix} + b\begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad \text{(AV\_1)}$$

With the normalization condition $aa^* + bb^* = 1$ (AV_2)

The relation between the classical components of the spin (directional cosines) and the coefficients a and b in of the spin wavefunction (AV_1) above is given by:

$$\vec{S} = \left(S_x,\ S_y,\ S_z\right) = \left(a^*,\ b^*\right)\left(\sigma_x,\ \sigma_y,\ \sigma_z\right)\begin{pmatrix} a \\ b \end{pmatrix} = \left(a^*,\ b^*\right)\left(\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix},\ \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}\ \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\right)\begin{pmatrix} a \\ b \end{pmatrix} =$$

$$\left(\left(a^*b + b^*a\right),\ -i\left(a^*b - b^*a\right),\ \left(a^*a - b^*b\right)\right) \quad \text{(AV\_3)}$$

The equation (AV_2) above with the equations (AV_4) to (AV_6) below determine the relation between the coefficients a, b of the spinor wavefunction and the classical directional cosines (S_x, S_y, S_z).

$$S_x = \left(a^*b + b^*a\right) \quad \text{(AV\_4)}$$

$$S_y = -i\left(a^*b - b^*a\right) \quad \text{(AV\_5)}$$

$$S_z = \left(a^*a - b^*b\right) \quad \text{(AV\_6)}$$

With no loss of generality is assumed that b is imaginary thus: b^*=-b

From Eq. (AV_2): $aa^* = 1 + b^2$ (AV_7)

Substituting (AV_7) into (AV_6) $S_z = \left(1 + 2b^2\right) \Rightarrow b = i\sqrt{\frac{1 - S_z}{2}} = i\frac{1 - S_z}{\sqrt{2\left(1 - S_z\right)}}$ (AV_8)

From Eq. (AV_4) and (AV_5)

$$S_x - iS_y = 2b^*a = -2ba = -2i\sqrt{\frac{1 - S_z}{2}}a \Rightarrow a = i\frac{S_x - iS_y}{\sqrt{2\left(1 - S_z\right)}} \quad \text{(AV\_9)}$$

$$\left.\begin{aligned} a &= i\frac{S_x - iS_y}{\sqrt{2\left(1-S_z\right)}} \\ b &= i\frac{1-S_z}{\sqrt{2\left(1-S_z\right)}} \end{aligned}\right\} \quad \text{(AV\_10)}$$

APPENDIX VI

This appendix contains the source of a fortran code to calculate the stable spin direction at the exit of a beam transfer line.

```
PROGRAM HELIX_RF
C*
        COMPLEX*16 SX(2,2),SY(2,2),SZ(2,2),ID(2,2)
        COMPLEX*16 IV(2,2),VAL
        COMPLEX*16
SRM1(2,2),SRM2(2,2),SRM3(2,2),SRM4(2,2),SRM5(2,2),SRM6(2,2),SRM7(2,2),SRM8(2,
2)
        COMPLEX*16
SRMH20(2,2),SRMK20(2,2),SRMA20(2,2),SRMB20(2,2),SRMC15(2,2),SRME20(2,2)
        COMPLEX*16
RMH20(2,2),RMK20(2,2),RMA20(2,2),RMB20(2,2),RMC15(2,2),RME20(2,2)
        COMPLEX*16 HR1(2,2),HR2(2,2),HR3(2,2),HR4(2,2),HR5(2,2),HR6(2,2)
        COMPLEX*16 HR(2,2),
MR1(2,2),MR2(2,2),MR3(2,2),MR4(2,2),MR5(2,2),MR6(2,2),MR7(2,2),MR8(2,2)
        COMPLEX*16 IHR(2,2),PM(2,2), RM(2,2)
        COMPLEX*16
RF1(2,2),RF2(2,2),RF3(2,2),RF4(2,2),RF5(2,2),RF6(2,2),RF7(2,2),RF8(2,2)
        COMPLEX*16 PM1(2,2),PM2(2,2),PM3(2,2),PM4(2,2)
        COMPLEX*16
PM5(2,2),PM6(2,2),PM7(2,2),PM8(2,2),PM9(2,2),PM10(2,2),PM11(2,2)
C
        COMPLEX*16 SRM_H10S(2,2), SRM_H11B(2,2), SRM_H12B(2,2), SRM_H13C(2,2)
        COMPLEX*16 SRM_UD01(2,2), SRM_UD02(2,2)
        COMPLEX*16 SRM_UD03(2,2), SRM_UD04(2,2), SRM_UD05(2,2), SRM_UD06(2,2)
        COMPLEX*16 SRM_SOL1(2,2), SRM_SOL2(2,2), SRM_SOL3(2,2), SRM_SOL4(2,2)
        COMPLEX*16 SRM_SOL5(2,2)
        COMPLEX*16 SRM_BMP1(2,2), SRM_BMP2(2,2), SRM_BMP3(2,2)
        COMPLEX*16 SRM_BMP1b(2,2),SRM_BMP2b(2,2),SRM_BMP3b(2,2)
        COMPLEX*16 SRM_WP01(2,2), SRM_WP02(2,2)
        COMPLEX*16 SRM_WD01(2,2), SRM_WD02(2,2), SRM_WD03(2,2), SRM_WD04(2,2)
        COMPLEX*16 SRM_WD05(2,2), SRM_WD06(2,2), SRM_WD07(2,2), SRM_WD08(2,2)
        COMPLEX*16 SRM_SWM(2,2),  SRM_WDerror(2,2)
        COMPLEX*16 SRM_G1D(2,2),  SRM_G2F(2,2), SRM_G3F(2,2), SRM_G4D(2,2),
SRM_G5D(2,2)
        COMPLEX*16 SRM_G6F(2,2),  SRM_G7F(2,2), SRM_G8D(2,2), SRM_G9D(2,2)
        COMPLEX*16 SRM_G10F(2,2), SRM_G11F(2,2), SRM_G12D(2,2), SRM_G13D(2,2)
        COMPLEX*16 SRM_G14F(2,2), SRM_G15F(2,2), SRM_G16D(2,2), SRM_G17D(2,2)
        COMPLEX*16 SRM_G18F(2,2), SRM_G19F(2,2), SRM_G20D(2,2), SRM_G21D(2,2)
        COMPLEX*16 SRM_G22F(2,2), SRM_G23F(2,2), SRM_G24D(2,2), SRM_G25D(2,2)
        COMPLEX*16 SRM_G26F(2,2), SRM_G27F(2,2), SRM_G28D(2,2), SRM_G29D(2,2)
        COMPLEX*16 SRM_G30F(2,2)
        COMPLEX*16 SRM_D31U(2,2), SRM_DP03(2,2), SRM_LAMB(2,2), SRM_RD01(2,2)
        COMPLEX*16 SRM_RKIK(2,2), SRM_Q01(2,2),  SRM_Q02(2,2)
C
        COMPLEX*16 FRM_2X2(2,2),TRM_2X2(2,2)
        REAL*8     FRM_3X3(3,3)
C
        REAL*8 V0,V1,V2,V3,V4,V5,PI,T,DT
        REAL*8 VF1,VF2,VF3,VF4,VF5,VF6,VF7,VF8
```

```
        REAL*8 RA,CX,CY,CZ,ZVAL,dir_cos
        REAL*8 Gv,gm0,Ggamma0,gm,Ggamma,gm_stp,Ggamma_stp
        REAL*8 eta1,eta2,eta3,eta4,eta5,eta6,eta7,eta8
        REAL*8 AH20, AK20, AA20, AB20, AC15, AE20
        REAL*8 AH20d,AK20d,AA20d,AB20d,AC15d,AE20d
        REAL*8 AS1,AS2,AS3,AS4,AS5,AS6,AS7,AS8
        REAL*8 AS1r,AS2r,AS3r,AS4r,AS5r,AS6r,AS7r,AS8r,A_step
        REAL*8 ddiff,ddiff1,ddiff2
        REAL*8 tv(20)
c
        Real*8 Sxi, Syi, Szi, Sxf, Syf, Szf, Sxf_max,Syf_max,Szf_max
        Real*8 S_ad, S_ar, eta,fdiff
c
        INTEGER*4 I,Nstp,IA,IA_stp
        CHARACTER*1 PR
        CHARACTER*20 fname
        CHARACTER*200 fline
        COMMON/PRINT/ PR
        COMMON/CONST/ PI
        COMMON/VFACT/ V1,V2,V3,V4,V5,V6
        COMMON/ZERO/ZVAL
C       X-RADIAL
C       Z-VERTICAL
C       Y-BEAM_DIRECTION
c
      write(2,*) ' '
      write(2,*)
     1   'Spinor in the X-direction: Positive x Point to the right '
      write(2,*) ' 0  1   '
      write(2,*) ' 1  0   '
      write(2,*)
     1   'Spinor in the y-direction:  Positive y-axis is along beam '
      write(2,*) ' 0 -i   '
      write(2,*) ' i  0   '
      write(2,*) 'Spinor in the z-direction: Positive z-axis is Up '
      write(2,*) ' 1  0   '
      write(2,*) ' 0 -1   '
c
c       write(*,*) ' Hit Return to continue '
c       read(*,*)
c
        PI=3.141592654
c
        amu0=0.93193827
        Amass=3.0
        Q0=2.0
c        Gv=1.7928
        Gv=-4.184
        p_mass=Amass*amu0
c
c       Initial Spin Direction
        Ggamma=65.5
        Sxi=0.0
        Syi=0.0
        Szi=1.0
c
```

```
c
       write(*,*) ' Make sure you have set the range of the Chicanes '
       write(*,*) ' '
c      read(*,*) ch1
c
       write(*,*) ' Enter Vertical Spin low and high limits '
       write(*,*) ' Results will be recorded above this limit '
c
        Szf_l_lim=0.85
        Szf_h_lim=0.999
c      read(*,*) Szf_l_lim, Szf_h_lim
c
       gamma=Ggamma/Gv
       P_mom=Amass*amu0*SQRT(gamma**2-1)
       Brho=3.3356*(P_mom/Q0)
c
c Solenoid #1
       Bf_Sol1=0.0
       Ln_Sol1=2.8
c
c Solenoid #2
       Bf_Sol2=0.0
       Ln_Sol2=2.8
c
c Solenoid #3
       Bf_Sol3=0.0
       Ln_Sol3=2.8
c
c Solenoid #4
       Bf_Sol4=-0.0
       Ln_Sol4=2.8
c
c Solenoid #5
       Bf_Sol5=-0.0
       Ln_Sol5=2.8
c
c       W_line Bump angle in Tesla WP02 Pitching
       S_a_W_Tesla=0.0
       W_length=1.0
       BL_WM1=S_a_W_Tesla*W_length
       BL_WM2=-2*BL_WM1
       BL_WM3=BL_WM1
c
c       Bump angle in degrees near RHIC Septum
       S_a_R_Tesla=0.0
       R_length=1.0
       BL_RM1=S_a_R_Tesla*R_length
       BL_RM2=-2*BL_RM1
       BL_RM3=BL_RM1
c
c       W_line Bump angle in Tesla WP01 Pitching
       S_a_WA_Tesla=0.0
       WA_length=1.0
       BL_WAM1=S_a_W_Tesla*WA_length
       BL_WAM2=-2*BL_WAM1
       BL_WAM3=BL_WAM1
```

```
c
C       CONSTRUCT SPINOR MATRICES
        CALL SPINORS(SX,SY,SZ,ID)
c
        write(*,*) ' '
        write(*,*) ' Before you run the code '
        write(*,*) ' Open the source file to change strengths of Bump&Solenoid
magnets '
        write(*,*) ' '
        write(*,*) ' Enter filename of initial Spin Direction in front of H10'
        write(*,*) ' Spin_Direction_at_H10'
c       read(*,'(a20)') fname
        fname='fort.1'
c
 1082   format( ' Filename of initial Spin Direction = ',A20)
        write(*,1082) fname
c       OPEN(UNIT=1,FILE=fname,STATUS='OLD')
        OPEN(UNIT=1,FILE='fort.1',STATUS='OLD')
c
c read first line of the file
        read(1, 123 , END=999) fline
c
        iprint=1
        iprint4=1
        nmax=3000
        ipr2=0
c
c
c Go over the Ggamma'a
c
        Do 1002 ns=1,nmax
c
        iflag_Spin=0
c
        read(1, 123 , END=999) fline
 123    format(A200)
c
c          write(*,123) fline
c
           read(fline,*)(tv(i),i=1,13)
c       write(*,*) tv(3),tv(6),tv(7),tv(8)
c
c
        Ggamma=tv(3)
c
c Check if Ggamma is half integer
        fdiff=ABS(ABS(INT(Ggamma)-Ggamma)-0.5)
        if(fdiff.gt.0.0001) go to 1002
c
c Initial spin direction
        Sxi= tv(6)
        Syi= tv(7)
        Szi= tv(8)
c
        gamma=Abs(Ggamma/Gv)
        P_mom=Amass*amu0*SQRT(gamma**2-1)
```

```
        Brho=3.3356*(P_mom/Q0)
        angle1=BL_WM1/Brho
        angle2=BL_RM1/Brho
c        write(*,103) Ggamma, Brho,1000*angle1,1000*angle2
 103    format(' Ggamma,Brho,ang1,ang2 ',4f8.2)
c
        ZVAL=0.0
c
c strength of 1st three Dipoles WP02 pitcing
        S_a_W_Tesla0=-2.0
        step_S_a_W_Tesla=0.05
        jd1_max=2*ABS(S_a_W_Tesla0)/step_S_a_W_Tesla+1
c       S_a_W_Tesla0=0.0
c       jd1_max=1
        BW_min=-S_a_W_Tesla0
c
c strength of 2nd set of three Dipoles 3 mrad pitching
        S_a_R_Tesla0=-2.0
        step_S_a_R_Tesla=0.1
        jd2_max=2*ABS(S_a_R_Tesla0)/step_S_a_R_Tesla+1
c       S_a_R_Tesla0=0.0
c       jd2_max=1
        BR_min=-S_a_R_Tesla0
c
        write(*,*) ' Ggamma=',Ggamma
cc
c Strength of 3-Bump magnets at WP01 pitching magnet Location #1.
        S_a_WA_Tesla0=-2.0
        step_S_a_WA_Tesla=0.05
        jd3_max=2*ABS(S_a_WA_Tesla0)/step_S_a_WA_Tesla+1
c        S_a_WA_Tesla0=0.0
c        jd3_max=1
        BWA_min=-S_a_WA_Tesla0
c
c       write(*,*) jd1_max,jd2_max,jd3_max
c*************************************************************
c*************************************************************
c
        write(4,*) '#'
        no=1
c
        Do 1001 jd1=1,jd1_max
c
c       W_line Bump angle in Tesla
        S_a_W_Tesla=S_a_W_Tesla0+(jd1-1)*step_S_a_W_Tesla
        W_length=1.0
        BL_WM1=S_a_W_Tesla*W_length
        BL_WM2=-2*BL_WM1
        BL_WM3=BL_WM1
c
c Go over the strength of 2nd set of three Dipoles
c
        Do 1000 jd2=1,jd2_max
c
c          write(*,*) ' ns, jd1, jd2 ' , ns, jd1,jd2
c
```

```
c       Match_line Bump angle in Tesla
        S_a_R_Tesla=S_a_R_Tesla0+(jd2-1)*step_S_a_R_Tesla
        R_length=1.0
        BL_RM1=S_a_R_Tesla*R_length
        BL_RM2=-2*BL_RM1
        BL_RM3=BL_RM1
c
cc Go over the strength of 1st three Dipoles WP1 pitching
c
        Do 1011 jd3=1,jd3_max
c
c       W_line Bump angle in Tesla
        S_a_WA_Tesla=S_a_WA_Tesla0+(jd3-1)*step_S_a_WA_Tesla
        WA_length=1.0
        BL_WAM1=S_a_WA_Tesla*WA_length
        BL_WAM2=-2*BL_WAM1
        BL_WAM3=BL_WAM1
c
c
c       write(*,*) jd1,jd2,jd3
c
        if(ipr2.eq.1)
     1  write(2,101) Ggamma, Sxi,Syi,Szi
 101    format('  Ggamma    Initial     Sx       Sy        Sz
',/,f8.3,10x,3f10.5)
c
c       H10 Septum angle  S_an=21.065 mrad=1.2069 deg
c Create Spin Rotation matrix for H10_Septum
c
c
c          write(*,*) ' Values of Ggamma, jd S_a_W_Tesla ' ,
Ggamma,jd,S_a_W_Tesla
c          read(*,*) iread
c
        if(ipr2.eq.1)
     1 write(2,*)
     1 ' H10Septum  Angle of bend to the right -1.17949 deg=20.586 mrad'
c     Angle of bend at H10_Septum
        S_ad=-1.179504
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
c
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_H10S,CX,CY,CZ,eta)
c       Calculate Final Matrix
        CALL MULT_MA(SRM_H10S,ID,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c
c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call Final_Spin_Direction
     1  (Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c
```

```
c**************************************** pause ' '
c
        if(ipr2.eq.1)
     1 write(2,*)
     1  ' H11B  Angle of bend to the left 0.678955 deg=-11.85 mrad'
c     Angle of bend at H11B fring field
        S_ad=0.678955
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_H11B,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_H11B,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c
c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c
c**********************************************************************

        if(ipr2.eq.1)
      1  write(2,*)
      1   'H12B  Angle of bend to the left -0.35423 deg=-6.1826 mrad '
c       Angle of bend at H12B fring field
        S_ad=0.3540879
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_H12B,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_H12B,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)

c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c
c**********************************************************************
c
        if(ipr2.eq.1)
     1  write(2,*) ' H13C  Angle of bend to the left -0.210427 deg=-3.6726
mrad '
c     Angle of bend at H13C fring field
        S_ad=0.2102755
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
```

```
         call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_H13C,CX,CY,CZ,eta)
         CALL MULT_MA(SRM_H13C,TRM_2X2,FRM_2X2)
         call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)

c        Convert SU(2) Rotation Matrix to SU(3)
         call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c        Final_Spin_Direction
         call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c
c

         if(ipr2.eq.1)
     1 write(2,*) ' UD01  Angle of bend to the right -2.15572018 deg '
c     Angle of bend at UD01
         S_ad=-2.156225
         S_ar=pi*(S_ad/180.0)
         eta=Ggamma*S_ar/2.0
         CX=0.0
         CY=0.0
         CZ=1.0
         call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_UD01,CX,CY,CZ,eta)
         CALL MULT_MA(SRM_UD01,TRM_2X2,FRM_2X2)
         call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)

c        Convert SU(2) Rotation Matrix to SU(3)
         call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c        Final_Spin_Direction
         call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

         if(ipr2.eq.1)
     1 write(2,*) ' UD02  Angle of bend to the right -2.15572018 deg  '
c     Angle of bend at UD02
         S_ad=-2.156225
         S_ar=pi*(S_ad/180.0)
         eta=Ggamma*S_ar/2.0
         CX=0.0
         CY=0.0
         CZ=1.0
         call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_UD02,CX,CY,CZ,eta)
         CALL MULT_MA(SRM_UD02,TRM_2X2,FRM_2X2)
         call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)

c        Convert SU(2) Rotation Matrix to SU(3)
         call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c        Final_Spin_Direction
         call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

         if(ipr2.eq.1)
     1 write(2,*) ' UD03  Angle of bend to the right -2.0 deg '
c     Angle of bend at UD03
         S_ad=-2.0
```

```
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_UD03,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_UD03,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)

c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

        if(ipr2.eq.1)
     1 write(2,*) ' UD04  Angle of bend to the right -2.0 deg '
c     Angle of bend at UD04
        S_ad=-2.0
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_UD04,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_UD04,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)

c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

        if(ipr2.eq.1)
     1 write(2,*) ' UD05  Angle of bend to the right -2.0 deg '
c     Angle of bend at UD05
        S_ad=-2.0
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_UD05,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_UD05,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)

c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c
```

```
        if(ipr2.eq.1)
     1 write(2,*) ' UD06  Angle of bend to the right -2.0 deg '
c     Angle of bend at UD06
        S_ad=-2.0
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_UD06,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_UD06,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)

c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

        if(ipr2.eq.1)
     1 write(2,*) ' WD01  Angle of bend to the right -2.499861 deg '
c     Angle of bend at WD01
        S_ad=-2.499861
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_WD01,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_WD01,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)

c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

        if(ipr2.eq.1)
     1 write(2,*) ' WD02  Angle of bend to the right -2.499861 deg '
c     Angle of bend at WD02
        S_ad=-2.499861
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_WD02,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_WD02,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
```

```
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c
        if(ipr2.eq.1)
     1 write(2,*) ' Solenoid_1 Spin Rotation Angle is given as input'
c
        eta=(1/2.0)*(1.0+Gv)*(Bf_Sol1*Ln_Sol1)/Brho
c
        CX=0.0
        CY=1.0
        CZ=0.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_SOL1,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_SOL1,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)

c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c
        S_ar=BL_WAM1/Brho
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_BMP1,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_BMP1,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c       call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c
c
        if(ipr2.eq.1)
     1 write(2,*) ' WP01 Pitching Angle of bend down -0.71619724 deg '
c     Angle of bend at WP01
        S_ad=-0.71619724
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=1.0
        CY=0.0
        CZ=0.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_WP01,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_WP01,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c       call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
```

```
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c
        if(ipr2.eq.1)
     1 write(2,*) ' Bump_2  Angle of bend pos or neg deg '
c     Angle of bend at Bump_2

        S_ar=BL_WAM2/Brho
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_BMP2,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_BMP2,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c       call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c
c

        if(ipr2.eq.1)
     1 write(2,*) ' Bump_3  Angle of bend pos or neg deg '
c     Angle of bend at Bump_3

        S_ar=BL_WAM3/Brho
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_BMP3,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_BMP3,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c       call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c
c
        if(ipr2.eq.1)
     1 write(2,*) ' Solenoid_3 Spin Rotation Angle is given as input'
        Bf_Sol3=0.0
        eta=(1/2.0)*(1.0+Gv)*(Bf_Sol3*Ln_Sol3)/Brho
c
        CX=0.0
        CY=1.0
        CZ=0.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_SOL3,CX,CY,CZ,eta)
```

```
        CALL MULT_MA(SRM_SOL3,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)

c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c
c
        if(ipr2.eq.1)
     1 write(2,*) ' WD03  Angle of bend to the right -2.499861 deg '
c     Angle of bend at WD03
        S_ad=-2.499861
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_WD03,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_WD03,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c
        if(ipr2.eq.1)
     1 write(2,*) ' WDerror  Angle of bend 0.015 deg '
c     Angle of bend of error
        S_ad=0.000
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_WDerror,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_WDerror,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

        if(ipr2.eq.1)
     1 write(2,*) ' WD04  Angle of bend to the right -2.499861 deg '
c     Angle of bend at WD04
        S_ad=-2.499861
```

```
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_WD04,CX,CY,CZ,eta)
       CALL MULT_MA(SRM_WD04,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
c       Convert SU(2) Rotation Matrix to SU(3)
       call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
       call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

       if(ipr2.eq.1)
     1 write(2,*) ' WD05  Angle of bend to the right -2.499861 deg '
c     Angle of bend at WD05
       S_ad=-2.499861
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_WD05,CX,CY,CZ,eta)
       CALL MULT_MA(SRM_WD05,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
c       Convert SU(2) Rotation Matrix to SU(3)
       call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
       call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c
c

       if(ipr2.eq.1)
     1 write(2,*) ' Solenoid_4 Spin Rotation Angle is given as input'
c
       Bf_Sol4=0.0
        eta=(1/2.0)*(1.0+Gv)*(Bf_Sol4*Ln_Sol4)/Brho
c
        CX=0.0
        CY=1.0
        CZ=0.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_SOL4,CX,CY,CZ,eta)
       CALL MULT_MA(SRM_SOL4,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)

c       Convert SU(2) Rotation Matrix to SU(3)
       call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
```

```
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c
c

        if(ipr2.eq.1)
     1 write(2,*) ' WD06  Angle of bend to the right -2.499861 deg '
c     Angle of bend at WD06
        S_ad=-2.499861
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_WD06,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_WD06,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

        if(ipr2.eq.1)
     1 write(2,*) ' WD07  Angle of bend to the right -2.499861 deg '
c     Angle of bend at WD07
        S_ad=-2.499861
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_WD07,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_WD07,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

        if(ipr2.eq.1)
     1 write(2,*) ' WD08  Angle of bend to the right -2.499861 deg '
c     Angle of bend at WD08
        S_ad=-2.499861
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
```

```
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_WD08,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_WD08,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c       call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c
c

        if(ipr2.eq.1)
     1 write(2,*) ' Solenoid_2 Spin Rotation Angle is given as input'
c
        Bf_Sol2=0.0
        eta=(1/2.0)*(1.0+Gv)*(Bf_Sol2*Ln_Sol2)/Brho
c
        CX=0.0
        CY=1.0
        CZ=0.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_SOL2,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_SOL2,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)

c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c
c

        if(ipr2.eq.1)
     1 write(2,*) ' Bump_1  Angle of bend pos or neg deg '
c     Angle of bend at Bump_1

c        S_ar=pi*(S_ad_B1/180.0)
        S_ar=BL_WM1/Brho
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_BMP1,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_BMP1,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c       call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c
```

```
        if(ipr2.eq.1)
     1 write(2,*) ' WP02 Pitching Angle of bend up 0.7167530 deg '
c     Angle of bend at WP02
        S_ad=0.7167530
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=1.0
        CY=0.0
        CZ=0.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_WP02,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_WP02,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)

c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

        if(ipr2.eq.1)
     1 write(2,*) ' Bump_2  Angle of bend pos or neg deg '
c     Angle of bend at Bump_2

        S_ar=BL_WM2/Brho
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_BMP2,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_BMP2,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c       call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c
c

        if(ipr2.eq.1)
     1 write(2,*) ' Bump_3  Angle of bend pos or neg deg '
c     Angle of bend at Bump_3

        S_ar=BL_WM3/Brho
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_BMP3,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_BMP3,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
```

```
c       call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

        if(ipr2.eq.1)
     1 write(2,*) ' Results in Front of SWM '
c
c       call Print_Results_at_Exit(Ggamma,Sxf,Syf,Szf,FRM_2X2)
c

        if(ipr2.eq.1)
     1 write(2,*) ' SWM Witching Angle of bend Right 2.74412149 '
c     Angle of bend at SWM
        S_ad=2.74412149
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_SWM,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_SWM,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
C
c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Dis_adrection
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c
        agrd1=2.75893027
        agrd2=2.22239626
c

        if(ipr2.eq.1)
     1 write(2,*) ' G1D Witching Angle of bend left 2.75893027 '
c     Angle of bend at D1D
        S_ad=agrd1
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_G1D,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_G1D,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
C       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
```

```
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

        if(ipr2.eq.1)
     1 write(2,*) ' G2F Witching Angle of bend left 2.22239626 '
c     Angle of bend at D2F
        S_ad=agrd2
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_G2F,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_G2F,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

        if(ipr2.eq.1)
     1 write(2,*) ' G3F Witching Angle of bend left 2.75893027 '
c     Angle of bend at D3F
        S_ad=agrd1
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_G3F,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_G3F,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

        if(ipr2.eq.1)
     1 write(2,*) ' G4D Witching Angle of bend left 2.75893027 '
c     Angle of bend at D4D
        S_ad=agrd1
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_G4D,CX,CY,CZ,eta)
```

```
        CALL MULT_MA(SRM_G4D,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
C       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

        if(ipr2.eq.1)
     1 write(2,*) ' G5D Witching Angle of bend left 2.75893027 '
c     Angle of bend at D5D
        S_ad=agrd1
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_G5D,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_G5D,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

        if(ipr2.eq.1)
     1 write(2,*) ' G6F Witching Angle of bend left 2.75893027 '
c     Angle of bend at D6F
        S_ad=agrd1
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_G6F,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_G6F,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

        if(ipr2.eq.1)
     1 write(2,*) ' G7F Witching Angle of bend left 2.75893027 '
c     Angle of bend at D7F
```

```
        S_ad=agrd1
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_G7F,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_G7F,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

        if(ipr2.eq.1)
     1 write(2,*) ' G8D Witching Angle of bend left 2.75893027 '
c     Angle of bend at G8D
        S_ad=agrd1
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_G8D,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_G8D,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

        if(ipr2.eq.1)
     1 write(2,*) ' G9D Witching Angle of bend left 2.75893027 '
c     Angle of bend at G9D
        S_ad=agrd1
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_G9D,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_G9D,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
```

```
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

        if(ipr2.eq.1)
     1 write(2,*) ' G10F Witching Angle of bend left 2.75893027 '
c     Angle of bend at G10F
        S_ad=agrd1
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_G10F,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_G10F,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

        if(ipr2.eq.1)
     1 write(2,*) ' G11F Witching Angle of bend left 2.75893027 '
c     Angle of bend at G11F
        S_ad=agrd1
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_G11F,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_G11F,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

        if(ipr2.eq.1)
     1 write(2,*) ' G12D Witching Angle of bend left 2.75893027 '
c     Angle of bend at G12D
        S_ad=agrd1
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_G12D,CX,CY,CZ,eta)
```

```
        CALL MULT_MA(SRM_G12D,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

        if(ipr2.eq.1)
     1 write(2,*) ' G13D Witching Angle of bend left 2.75893027 '
c     Angle of bend at G13D
        S_ad=agrd1
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_G13D,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_G13D,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

        if(ipr2.eq.1)
     1 write(2,*) ' G14F Witching Angle of bend left 2.75893027 '
c     Angle of bend at D14F
        S_ad=agrd1
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_G14F,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_G14F,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

        if(ipr2.eq.1)
     1 write(2,*) ' G15F Witching Angle of bend left 2.75893027 '
c     Angle of bend at D15F
```

```
        S_ad=agrd1
         S_ar=pi*(S_ad/180.0)
         eta=Ggamma*S_ar/2.0
         CX=0.0
         CY=0.0
         CZ=1.0
         call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_G15F,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_G15F,TRM_2X2,FRM_2X2)
         call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c         call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

        if(ipr2.eq.1)
     1 write(2,*) ' G16D Witching Angle of bend left 2.75893027 '
c     Angle of bend at D16D
        S_ad=agrd1
         S_ar=pi*(S_ad/180.0)
         eta=Ggamma*S_ar/2.0
         CX=0.0
         CY=0.0
         CZ=1.0
         call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_G16D,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_G16D,TRM_2X2,FRM_2X2)
         call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c         call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
C       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

        if(ipr2.eq.1)
     1 write(2,*) ' G17D Witching Angle of bend left 2.75893027 '
c     Angle of bend at D17D
        S_ad=agrd1
         S_ar=pi*(S_ad/180.0)
         eta=Ggamma*S_ar/2.0
         CX=0.0
         CY=0.0
         CZ=1.0
         call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_G17D,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_G17D,TRM_2X2,FRM_2X2)
         call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c         call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
C       call Rotation_matrix_of_Elements(FRM_2X2,1)
c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
```

```
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

        if(ipr2.eq.1)
     1 write(2,*) ' G18F Witching Angle of bend left 2.75893027 '
c     Angle of bend at D18F
        S_ad=agrd1
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_G18F,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_G18F,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

        if(ipr2.eq.1)
     1 write(2,*) ' G19F Witching Angle of bend left 2.75893027 '
c     Angle of bend at D19F
        S_ad=agrd1
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_G19F,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_G19F,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
C      Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

        if(ipr2.eq.1)
     1 write(2,*) ' G20D Witching Angle of bend left 2.75893027 '
c     Angle of bend at D20D
        S_ad=agrd1
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
```

```
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_G20D,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_G20D,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

        if(ipr2.eq.1)
     1 write(2,*) ' G21D Witching Angle of bend left 2.75893027 '
c     Angle of bend at D21D
        S_ad=agrd1
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_G21D,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_G21D,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

        if(ipr2.eq.1)
     1 write(2,*) ' G22F Witching Angle of bend left 2.75893027 '
c     Angle of bend at D22F
        S_ad=agrd1
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_G22F,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_G22F,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
C       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

        if(ipr2.eq.1)
     1 write(2,*) ' G23F Witching Angle of bend left 2.75893027 '
```

```
c     Angle of bend at D23F
        S_ad=agrd1
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_G23F,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_G23F,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

        if(ipr2.eq.1)
     1 write(2,*) ' G24D Witching Angle of bend left 2.75893027 '
c     Angle of bend at D24D
        S_ad=agrd1
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_G24D,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_G24D,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c
        if(ipr2.eq.1)
     1 write(2,*) ' G25D Witching Angle of bend left 2.75893027 '
c     Angle of bend at D25D
        S_ad=agrd1
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_G25D,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_G25D,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
```

```
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

        if(ipr2.eq.1)
     1 write(2,*) ' G26F Witching Angle of bend left 2.75893027 '
c     Angle of bend at D26F
        S_ad=agrd1
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_G26F,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_G26F,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

        if(ipr2.eq.1)
     1 write(2,*) ' G27F Witching Angle of bend left 2.22239626 '
c     Angle of bend at D27F
        S_ad=agrd2
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_G27F,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_G27F,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
C       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

        if(ipr2.eq.1)
     1 write(2,*) ' G28D Witching Angle of bend left 2.22239626 '
c     Angle of bend at D28D
        S_ad=agrd2
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_G28D,CX,CY,CZ,eta)
```

```
        CALL MULT_MA(SRM_G28D,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
C       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

        if(ipr2.eq.1)
     1 write(2,*) ' G29D Witching Angle of bend left 2.22239626 '
c     Angle of bend at D29D
        S_ad=agrd2
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_G29D,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_G29D,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
C       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

        if(ipr2.eq.1)
     1 write(2,*) ' G30F Witching Angle of bend left 2.22239626 '
c     Angle of bend at G30F
        S_ad=agrd2
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_G30F,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_G30F,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
cc        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

        if(ipr2.eq.1)
     1 write(2,*) ' D31U Witching Angle of bend left 2.45233282 '
c     Angle of bend at D31U
```

```
        S_ad=2.45233282
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_D31U,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_D31U,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
cc        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
C       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

        if(ipr2.eq.1)
     1 write(2,*) ' Solenoid_5 Spin Rotation Angle is given as input'
c
        eta=(1/2.0)*(1.0+Gv)*(Bf_Sol5*Ln_Sol5)/Brho
c
        CX=0.0
        CY=1.0
        CZ=0.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_SOL5,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_SOL5,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)

c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
cc

        if(ipr2.eq.1)
     1 write(2,*) ' Bump_1b  Angle of bend pos or neg deg '
c     Angle of bend at Bump_1b

        S_ar=BL_RM1/Brho
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_BMP1b,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_BMP1b,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c       call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
```

```
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

        if(ipr2.eq.1)
     1 write(2,*)' DP03 Pitching Witching Angle of bend down -0.1718873'
c     Angle of bend at DP03
        S_ad=-0.18718873
         S_ar=pi*(S_ad/180.0)
         eta=Ggamma*S_ar/2.0
         CX=1.0
         CY=0.0
         CZ=0.0
         call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_DP03,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_DP03,TRM_2X2,FRM_2X2)
         call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
cc         call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
C       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c
c

        if(ipr2.eq.1)
     1 write(2,*) ' Bump_2b  Angle of bend pos or neg deg '
c     Angle of bend at Bump_2b

        S_ar=BL_RM2/Brho
         eta=Ggamma*S_ar/2.0
         CX=0.0
         CY=0.0
         CZ=1.0
         call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_BMP2b,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_BMP2b,TRM_2X2,FRM_2X2)
         call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c
c

        if(ipr2.eq.1)
     1 write(2,*) ' Bump_3b  Angle of bend pos or neg deg '
c     Angle of bend at Bump_3b

        S_ar=BL_RM3/Brho
         eta=Ggamma*S_ar/2.0
         CX=0.0
         CY=0.0
```

```
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_BMP3b,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_BMP3b,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c       call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

        if(ipr2.eq.1)
     1 write(2,*) ' Results in Front of SWM '
c
c       call Print_Results_at_Exit(Ggamma,Sxf,Syf,Szf,FRM_2X2)
c

        if(ipr2.eq.1)
     1 write(2,*) ' LAMB Angle of bend lefr 2.1772282 deg '
c     Angle of bend at LAMB
        S_ad=2.1772282
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_LAMB,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_LAMB,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
cc        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

        if(ipr2.eq.1)
     1 write(2,*) ' RQ01 Pitching Angle of bend up 0.11688 deg'
c     Angle of bend at QUAD01
        S_ad=0.13061688
c     Angle of bend at DP03
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=1.0
        CY=0.0
        CZ=0.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_Q01,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_Q01,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
cc        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
c       Convert SU(2) Rotation Matrix to SU(3)
```

```
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

        if(ipr2.eq.1)
     1 write(2,*) ' RD01  Angle of bend right -2.28380 '
c     Angle of bend at RD01
        S_ad=-2.2298380
c     Angle of bend at DP03
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=0.0
        CY=0.0
        CZ=1.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_RD01,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_RD01,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
cc        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

        if(ipr2.eq.1)
     1 write(2,*) ' RQ02 Pitching Angle of bend down -0.044118 deg'
c     Angle of bend at QUAD01
        S_ad=-0.0469118
c     Angle of bend at DP03
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
        CX=1.0
        CY=0.0
        CZ=0.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_Q02,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_Q02,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
cc        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
c       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c

        if(ipr2.eq.1)
     1 write(2,*) ' RKIK Pitching Angle of bend up 0.103 '
c     Angle of bend at RKIK
        S_ad=0.103
        S_ar=pi*(S_ad/180.0)
        eta=Ggamma*S_ar/2.0
```

```
        CX=1.0
        CY=0.0
        CZ=0.0
        call Spin_Rot_Mat(ID,SX,SY,SZ,SRM_RKIK,CX,CY,CZ,eta)
        CALL MULT_MA(SRM_RKIK,TRM_2X2,FRM_2X2)
        call Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
cc        call Check_Rotation_matrix_of_Elements(FRM_2X2, iflag)
c
C       Convert SU(2) Rotation Matrix to SU(3)
        call SU2_to_SU3(FRM_2X2,FRM_3X3,1)
c       Final_Spin_Direction
        call
Final_Spin_Direction(Sxf,Syf,Szf,FRM_3X3,Ggamma,Sxi,Syi,Szi,S_ar,ipr2)
c
c       write(2,*) ' Final results ',Sxf,Syf,Szf
c       write(*,*) ' Final results ',jd1,jd2,jd3,Sxf,Syf,Szf
c
c       assign a maximum value to the spin SzF directions
        if(jd1+jd2+jd3.eq.3)  then
           Sxf_max=Sxf
           Syf_max=Syf
           Szf_max=Szf
        Go to 1011
        endif
c
c       Check if the absolute vertical spin is withing the set limits
        if((ABS(Szf).ge.Szf_l_lim).and.(ABS(Szf).le.Szf_h_lim)) then
c
c       Check if the new spin is larger than the previous spin
        if(ABS(Szf).gt.ABS(Szf_max)) then
           B1_max=S_a_W_Tesla
           B2_max=S_a_R_Tesla
           B3_max=S_a_WA_Tesla
c
           if((BW_min.ge.ABS(S_a_W_Tesla)).or.(BR_min.ge.ABS(S_a_R_Tesla)).or.
     1     (BR_min.ge.ABS(S_a_WA_Tesla))) then
            BW_min=ABS(S_a_W_Tesla)
            BR_min=ABS(S_a_R_Tesla)
            BWA_min=ABS(S_a_WA_Tesla)
c           write(7,*) no,BW_min,BR_min,BWA_min,Szf_max
            npp=no
           endif
c
           Sxf_max=Sxf
           Syf_max=Syf
           Szf_max=Szf
           iflag_Spin=1
           nos=no+1
           endif
        endif
c
c       if(iprint.eq.1) write(*,*) iprint
c
        if((ABS(Szf).ge.Szf_l_lim)) then
          no = no+1
c
```

```
        call Print_Res_at_Ex4(no,Ggamma,Sxf,Syf,Szf,FRM_2X2,iprint4,
     1                        S_a_W_Tesla,S_a_R_Tesla,S_a_WA_Tesla)
c
c       write(*,*) 'Szf= ',Szf
c       read(*,*)
        endif
c
 1011   continue
c
 1000   continue
c
 1001   continue
c
        write(7,222) npp, Ggamma, BW_min,BR_min,BWA_min,Szf_max
 222    format(I5,5f10.4)
        if(iflag_Spin.eq.1) then
         call Print_Res_at_Ex3(nos,Ggamma,Sxf_max,Syf_max,Szf_max,
     1                        FRM_2X2,iprint,B1_max,B2_max,B3_max)
        endif
c
 1002   continue
c
c
 999    continue
        close(1)
        Close(3)
c
        write(*,*)
     1  ' Output Spin Direction at RHIC in files fort.3 and fort.4'
c
2000    CONTINUE

        END

        SUBROUTINE SPINORS(SX,SY,SZ,ID)
C*
        COMPLEX*16 SX(2,2),SY(2,2),SZ(2,2),ID(2,2)
c
C        SX ... RADIAL_DIRECTION
        SX(1,1)=DCMPLX(0,0)
        SX(1,2)=DCMPLX(1,0)
        SX(2,1)=DCMPLX(1,0)
        SX(2,2)=DCMPLX(0,0)

C        SY ... BEAM_DIRECTION
        SY(1,1)=DCMPLX(0,0)
        SY(1,2)=DCMPLX(0,-1)
        SY(2,1)=DCMPLX(0,1)
        SY(2,2)=DCMPLX(0,0)

C        SZ ... VERTICAL_DIRECTION
        SZ(1,1)=DCMPLX(1,0)
        SZ(1,2)=DCMPLX(0,0)
        SZ(2,1)=DCMPLX(0,0)
        SZ(2,2)=DCMPLX(-1,0)
```

```
C       ID ... IDENTITY_MATRIX
        ID(1,1)=DCMPLX(1,0)
        ID(1,2)=DCMPLX(0,0)
        ID(2,1)=DCMPLX(0,0)
        ID(2,2)=DCMPLX(1,0)

C       WRITE(2,10) SX(1,1),SX(1,2),SX(2,1),SX(2,2)
C       WRITE(2,10) SY(1,1),SY(1,2),SY(2,1),SY(2,2)
C       WRITE(2,10) SZ(1,1),SZ(1,2),SZ(2,1),SZ(2,2)
C       WRITE(2,10) ID(1,1),ID(1,2),ID(2,1),ID(2,2)
10      FORMAT(2X,2F4.0,2X,2F4.0,/)

        RETURN
        END

        SUBROUTINE IDMULT(IM,VALUE,IV)

        COMPLEX*16 IM(2,2),IV(2,2),VALUE

        IV(1,1)=VALUE*IM(1,1)
        IV(1,2)=VALUE*IM(1,2)
        IV(2,1)=VALUE*IM(2,1)
        IV(2,2)=VALUE*IM(2,2)

c       WRITE(2,10) IM(1,1),IM(1,2),IM(2,1),IM(2,2)
c       WRITE(2,10) IV(1,1),IV(1,2),IV(2,1),IV(2,2)
10      FORMAT(2X,2F8.4,2X,2F8.4,/)
c       PAUSE ' CONT'

        RETURN
        END



C*************************************************************
        SUBROUTINE Spin_Rot_Mat(ID,SX,SY,SZ,SRMat,CX,CY,CZ,SRAng)

        COMPLEX*16 SX(2,2),SY(2,2),SZ(2,2),ID(2,2)
        COMPLEX*16 SRMat(2,2)
         REAL*8     SRAng
        REAL*8     CX,CY,CZ,PI
c
        CHARACTER*1 PRNT
        COMMON/PRINT/ PRNT
        COMMON/CONST/ PI
c
c       write(2,*) ' Rot Angle in Rad ', SRang
c
C     Generate Spin Rotation Matrix
        CALL GEN_SROTM(ID,SX,SY,SZ,SRMat,CX,CY,CZ,SRAng)
        call Print_Rotation_matrix_of_Elements(SRMat, iflag)
         call Check_Rotation_matrix_of_Elements(SRMat, iflag)
        RETURN
        END
C*************************************************************************
*
```

```
      SUBROUTINE GEN_SROTM(ID,SX,SY,SZ,SRMat,CX,CY,CZ,SRAng)

      COMPLEX*16 SX(2,2),SY(2,2),SZ(2,2),ID(2,2)
      COMPLEX*16 SRMat(2,2)
      REAL*8     SRAng

      COMPLEX*16 RMC(2,2),RMS(2,2) ,RMA(2,2)
      COMPLEX*16 RMSX(2,2),RMSY(2,2),RMSZ(2,2),RMS1(2,2),VALUE
      REAL*8     CX,CY,CZ,VL,RA,RAV1,ZVAL
      COMMON/ZERO/ZVAL

C     WRITE(2,*) ' ID '

c      convert SRAng into complex number
      VALUE=DCMPLX(DCOS(SRAng),ZVAL)
c      WRITE(2,*) ' VALUE ', VALUE,SRAng
c
C     MULTIPLY (IDENTITY MATRIX)*COS(SRAng)
      CALL IDMULT(ID,VALUE,RMC)
c
c     WRITE(2,10) RMC(1,1),RMC(1,2),RMC(2,1),RMC(2,2)
10    FORMAT(2X,2F8.4,2X,2F8.4,/)
c     PAUSE 'CONT'

      VL=DSQRT(CX*CX+CY*CY+CZ*CZ)
C     WRITE(2,*) ' Spinor SX '

      VALUE=DCMPLX(CX,ZVAL)
C     MULTIPLY (SX SPINNOR MATRIX)*CX
      CALL IDMULT(SX,VALUE,RMSX)

C     WRITE(2,*) ' Spinor SY '

C     MULTIPLY (SX SPINNOR MATRIX)*CY
      VALUE=DCMPLX(CY,ZVAL)
      CALL IDMULT(SY,VALUE,RMSY)

C     WRITE(2,*) ' Spinor SZ '

C     MULTIPLY (SX SPINNOR MATRIX)*CZ
      VALUE=DCMPLX(CZ,ZVAL)
      CALL IDMULT(SZ,VALUE,RMSZ)

C     CONSTRUCT PART OF FINAL ROTATION MATRIX
      RMS1(1,1)=(RMSX(1,1)+RMSY(1,1)+RMSZ(1,1))/VL
      RMS1(1,2)=(RMSX(1,2)+RMSY(1,2)+RMSZ(1,2))/VL
      RMS1(2,1)=(RMSX(2,1)+RMSY(2,1)+RMSZ(2,1))/VL
      RMS1(2,2)=(RMSX(2,2)+RMSY(2,2)+RMSZ(2,2))/VL

      VALUE=DCMPLX(ZVAL,DSIN(SRAng))
      CALL IDMULT(RMS1,VALUE,RMS)

C     CONSTRUCT FINAL MATRIX FOR THIS ELEMENT
      SRMat(1,1)=RMC(1,1)-RMS(1,1)
      SRMat(1,2)=RMC(1,2)-RMS(1,2)
      SRMat(2,1)=RMC(2,1)-RMS(2,1)
```

```
        SRMat(2,2)=RMC(2,2)-RMS(2,2)

        RETURN
        END

C*****************************************************************
        SUBROUTINE MULT_MA(A,B,AB)

C       MULTIPLIES A.B=AB MATRICES

        COMPLEX*16 A(2,2),B(2,2),AB(2,2)

        AB(1,1)=A(1,1)*B(1,1)+A(1,2)*B(2,1)
        AB(1,2)=A(1,1)*B(1,2)+A(1,2)*B(2,2)
        AB(2,1)=A(2,1)*B(1,1)+A(2,2)*B(2,1)
        AB(2,2)=A(2,1)*B(1,2)+A(2,2)*B(2,2)

        RETURN
        END

        SUBROUTINE INVERSE(A,IA)

        COMPLEX*16 A(2,2),IA(2,2),DET
        COMPLEX*16 PM(2,2)

        DET=(-(A(1,2)*A(2,1))+(A(1,1)*A(2,2)))

        IA(1,1)=A(2,2)/DET
        IA(1,2)=-A(1,2)/DET
        IA(2,1)=-A(2,1)/DET
        IA(2,2)=A(1,1)/DET

        CALL MULT_MA(A,IA,PM)
C       WRITE(2,10) PM(1,1),PM(1,2),PM(2,1),PM(2,2)
10      FORMAT(2X,2F9.5,2X,2F9.5,/)
C       PAUSE ' INVERSE CONT'


        RETURN
        END
c*****************************************************
        SUBROUTINE CALCUL(N,NA,T,ID,SX,SY,SZ,RM,IHR,Ggamma,Ggamma_stp)
c
        COMPLEX*16 SX(2,2),SY(2,2),SZ(2,2),IHR(2,2)
        COMPLEX*16 PX(2,2),PY(2,2),PZ(2,2)
        COMPLEX*16 TX1,TY1,TZ1
        COMPLEX*16 PR,TRACE,TX,TY,TZ
        COMPLEX*16 ID(2,2),RM(2,2), V(2,2)
        REAL*8  PI,COSR,SINR,ACOSR,ROTAD,CV2,T,ZVAL,rsmod
        REAL*8  COSX,COSY,COSZ
        REAL*8  ANGX,ANGY,ANGZ,ANGXY
        REAL*8  V1,V2,V3,V4,V5
        INTEGER*4 N
c
         REAL*8 cospnue,pnue,tnue,gm,Ggamma,ratio,remn,rI_ratio
        REAL*8 tfcos,fcos,rfcos,ficos,rficos,Ggamma_stp,gpr,ddiff
```

```
      REAL*8 fnue,tfnue,ctfnue,rfnue,trfnue,ctrfnue
       REAL*8 two, proper_sptun, proper_Isptun, cos_pro_Isptun
      real*8 Half_Trace
      COMMON/CONST/ PI
      COMMON/VFACT/ V1,V2,V3,V4,V5,V6
      COMMON/ZERO/ZVAL
C
C       write(2,*) ' Ggamma fractional tune', N,Ggamma, fnue
C
      two=2.0
C
      CALL MULT_MA(ID,RM,V)

C     V IS THE FINAL ROTATION MATRIX

      TRACE=V(1,1)+V(2,2)

      COSR=0.5*DREAL(TRACE)
      Half_Trace = 0.5*DREAL(TRACE)
       cospnue=COSR
       pnue=DACOS(COSR)
       fnue = pnue/pi
C
C       write(2,*) ' Ggamma fractional tune', N,Ggamma, fnue
C
C     ANGLE OF ROTATION ANGLE
      ACOSR=2.0*DACOS(COSR)
      ROTAD=(180.0/PI)*(ACOSR)
C
C     DIRECTIONAL COSINE
      SINR=DSQRT(1.-COSR*COSR)
      CV2=2.0*SINR
      PR=DCMPLX(ZVAL,CV2)
      CALL MULT_MA(SX,V,PX)
      TX1=PX(1,1)+PX(2,2)
      TX=TX1/PR
C     COSX X-DIRECTIONAL COSINE OF AXIS OF ROTATION
      COSX=DREAL(TX)
C     WRITE(2,101) TX,COSX
101   FORMAT(1X, 'P1',2F10.4)

      ANGX=(180.0/PI)*DACOS(COSX)
C
      CALL MULT_MA(SY,V,PY)
      TY1=PY(1,1)+PY(2,2)
      TY=TY1/PR
C     COSY Y-DIRECTIONAL COSINE OF AXIS OF ROTATION
      COSY=DREAL(TY)
      ANGY=(180.0/PI)*DACOS(COSY)
C
      CALL MULT_MA(SZ,V,PZ)
      TZ1=PZ(1,1)+PZ(2,2)
      TZ=TZ1/PR
C     COSZ Z-DIRECTIONAL COSINE OF AXIS OF ROTATION
      COSZ=DREAL(TZ)
      ANGZ=(180.0/PI)*DACOS(COSZ)
```

```
        rsmod=dsqrt(COSX*COSX+COSY*COSY+COSZ*COSZ)
c
        rfnue=1.0-fnue
c
        IGgamma=Ggamma
c
          remn=DMOD(Dfloat(IGgamma),two)

c
           tfnue=IGgamma+fnue
           ctfnue=DCOS(pi*tfnue)
c
           proper_sptun=  fnue
           proper_Isptun=tfnue
c
           trfnue=IGgamma+rfnue
           ctrfnue=DCOS(pi*trfnue)
c
           if(remn.gt.0.1) then
              proper_sptun=rfnue
              proper_Isptun=trfnue
           endif
c
           cos_pro_Isptun=DCOS(pi*proper_Isptun)
           ACOSR=2.0*pi*proper_Isptun
c
        write(4,12) N, Ggamma,Half_Trace, fnue, tfnue, ctfnue,
     1                                    rfnue, trfnue,ctrfnue,
     1              proper_sptun,proper_Isptun,cos_pro_Isptun
 12     format(I8,2x,F10.6,2x,F12.7,9F12.6)
c
        WRITE(1,20) N,NA,Ggamma,Half_Trace,ACOSR,COSX,COSY,COSZ,
     1  ANGX,ANGY,ANGZ
c
20      FORMAT(1X,I6,1x,I6,F14.7,F16.7,F13.7,4F13.6,4F12.5)
c
        WRITE(3,1)
1       FORMAT('#     N    Ggamma  Rot-ANG',5X,'dirCX',5X,'dirCY'
     1  ,5x,'dirCZ',5X,'ANGX',7X,'ANGY',5X,'ANGZ'
     1  ,7X,' MOD')
        WRITE(3,21) N,Ggamma,ROTAD,COSX,COSY,COSZ,ANGX,ANGY,ANGZ,rsmod
c
 21     FORMAT(I6,F9.5,F10.3,3F10.4,3F10.3,F10.4,/)

        RETURN
        END
C*****************************************************************
c
        Subroutine SU2_to_SU3(FRM, FRM_3x3,iflag)
c
        COMPLEX*16 FRM(2,2)
        REAL*8     FRM_3X3(3,3)
c
c       Check Rotation Matrix
c
```

```
c
        FRM_3X3(1,1)= DREAL( FRM(1,1)*FRM(1,1)-FRM(1,2)*FRM(1,2) )
        FRM_3X3(1,2)= DIMAG( FRM(1,1)*FRM(1,1)+FRM(1,2)*FRM(1,2) )
        FRM_3X3(1,3)=-2.0*DREAL( FRM(1,1)*FRM(1,2) )
c
        FRM_3X3(2,1)=-DIMAG( FRM(1,1)*FRM(1,1)-FRM(1,2)*FRM(1,2) )
        FRM_3X3(2,2)= DREAL( FRM(1,1)*FRM(1,1)+FRM(1,2)*FRM(1,2) )
        FRM_3X3(2,3)= 2.0*DIMAG( FRM(1,1)*FRM(1,2) )
c
        FRM_3X3(3,1)=2.0*DREAL( FRM(1,1)*DCONJG(FRM(1,2)) )
        FRM_3X3(3,2)=2.0*DIMAG( FRM(1,1)*DCONJG(FRM(1,2)) )
        FRM_3X3(3,3)= FRM(1,1)*DCONJG(FRM(1,1))-FRM(1,2)*DCONJG(FRM(1,2))
c
        call Print_3X3_Mat(FRM_3X3,iflag)
c
        RETURN
        END
C*************************************************************************
c
        subroutine Check_Rotation_Matrix(FRM,S_ar)
c
        COMPLEX*16 FRM(2,2)
        Real*8 S_ar
c
        write(2,*) ' '
        write(2,*) 'Check Spinor Rotation Matrix '
        write(2,*) ' '
        Write(2,*)  FRM(1,1), FRM(1,2)
        Write(2,*)  FRM(2,1), FRM(2,2)
        write(2,*) ' '
c
        return
        end
c*********************************************************
c
        Subroutine Final_Spin_Direction
     1  (Sxf,Syf,Szf,FRM_3X3,Gg,Sxi,Syi,Szi,S_ar,ipr2)
c
        REAL*8   Sxf,Syf,Szf,  FRM_3X3(3,3), Sxi,Syi,Szi, S_ar, Spin, Gg
c
           Sxf= FRM_3X3(1,1)*Sxi+FRM_3X3(1,2)*Syi+FRM_3X3(1,3)*Szi
           Syf= FRM_3X3(2,1)*Sxi+FRM_3X3(2,2)*Syi+FRM_3X3(2,3)*Szi
           Szf= FRM_3X3(3,1)*Sxi+FRM_3X3(3,2)*Syi+FRM_3X3(3,3)*Szi
c
           Spin=DSQRT(Sxf*Sxf+Syf*Syf+Szf*Szf)
c       write(2,*) ' '
c       write(2,*) ' Initial Spin Dir. Cosines ', Sxi,Syi,Szi

        if(ipr2.eq.1)
     1 write(2,101) Gg, Sxf,Syf,Szf ,Spin
 101    format('  Ggamma      Sx        Sy        Sz
Spin',/,f8.3,4f10.5)
        if(ipr2.eq.1)
     1 write(2,*) ' '
c
        return
```

```
      end
C*********************************************************************
c
      Subroutine Print_Rotation_matrix_of_Elements(RME, iflag)
c
      COMPLEX*16 RME(2,2)
c
c     write(2,*) ' '
      if(ipr2.eq.1) then
      if(iflag.eq.0) write(2,*) 'Spinor Rotation Matrix of Element'
      if(iflag.eq.1) write(2,*) 'Spinor Rotation Matrix of all Element'
      endif
c     write(2,*) ' '
c     Write(2,*)  RME(1,1), RME(1,2)
c     Write(2,*)  RME(2,1), RME(2,2)
c     write(2,*) ' '
c
      return
      end
C*********************************************************************
c
      Subroutine Check_Rotation_matrix_of_Elements(FRM, iflag)
c
      COMPLEX*16 FRM(2,2), FMat_MOD
c
c
c     write(2,*) ' Check Rotation Matrix '
c     write(2,*) ' Complex_Conj(FRM(1,1)) = FRM(2,2) '
c     write(2,*) ' Complex_Conj(FRM(1,1))= ', DCONJG(FRM(1,1))
c     write(2,*) '              (FRM(2,2))= ',(FRM(2,2))
c     write(2,*) ' '
c
c     write(2,*) '-Complex_Conj(FRM(1,2)) = FRM(2,1) '
c     write(2,*) '-Complex_Conj(FRM(1,2))= ',-DCONJG(FRM(1,2))
c     write(2,*) '              (FRM(2,1))= ',(FRM(2,1))
c     write(2,*) ' '
      FMat_MOD= FRM(1,1)*FRM(2,2)-FRM(2,1)*FRM(1,2)
c     write(2,*) ' Determinant ' , FMat_MOD
c     write(2,*) ' '

c
c
c     write(2,*) ' '
c
      return
      end
C******************
c*********************************************************************
c*********************************************************************
c
      Subroutine Print_3X3_Mat(FRM_3X3,iflag)
c
      Real*8  FRM_3X3(3,3)
      if(iflag.eq.0) write(2,*) ' 3X3 Rotation Matrix of the Element '
c     if(iflag.eq.1) write(2,*) ' 3X3 Rotation Matrix of all Elements '
c     Write(2,102)  FRM_3X3(1,1), FRM_3X3(1,2), FRM_3X3(1,3)
```

```
c       Write(2,102)  FRM_3X3(2,1), FRM_3X3(2,2), FRM_3X3(2,3)
c       Write(2,102)  FRM_3X3(3,1), FRM_3X3(3,2), FRM_3X3(3,3)
c       write(2,*) ' '
 102    FORMAT(3F16.7)
c
        return
        end
C*********************************************************************
C*********************************************************************

        Subroutine Total_Rotation_Matrix(FRM_2X2,TRM_2X2)
c
        COMPLEX*16 FRM_2X2(2,2), TRM_2X2(2,2)
c
        TRM_2X2(1,1)=FRM_2X2(1,1)
        TRM_2X2(1,2)=FRM_2X2(1,2)
        TRM_2X2(2,1)=FRM_2X2(2,1)
        TRM_2X2(2,2)=FRM_2X2(2,2)
c
        return
        end
c*********************************************************************
c
        subroutine Print_Res_at_Ex3
     1  (ns,Ggamma,Sxf,Syf,Szf,FRM_2X2,iprint,Bf_1st,Bf_2nd,Bf_3rd)
c
        COMPLEX*16 FRM_2X2(2,2)
        COMPLEX*16 TRACE
c
        REAL*8     pi, COSR, ACOSR, ROTAD,Sxf,Syf,Szf,Spin,Ggamma
        REAL*8     ANGX,ANGY,ANGZ
c
        pi=3.141592654
c
        TRACE=FRM_2X2(1,1)+FRM_2X2(2,2)
        COSR=0.5*DREAL(TRACE)
        ACOSR=2.0*DACOS(COSR)
        ROTAD=(180.0/PI)*(ACOSR)
c
        ANGX=(180.0/PI)*DACOS(Sxf)
        ANGy=(180.0/PI)*DACOS(Syf)
        ANGz=(180.0/PI)*DACOS(Szf)
c
           Spin=DSQRT(Sxf*Sxf+Syf*Syf+Szf*Szf)
c
c       if(iprint.eq.1) WRITE(*,1)
1       FORMAT('# Final results ',/,
     1  '#     ns   Ggamma   Rot-ANG',5X,'cosSx',5X,'cosSy'
     1  ,5x,'cosSz',5X,'ANGX',7X,'ANGY',5X,'ANGZ'
     1  ,6X,' MOD',3x,' Bfld_1st',2x,' Bfld_2nd',2x,' Bfld_3rd',
     1   3x,'|Szf|'
     1  ,3x,'|Bf1st|',3x,'|Bf2nd|',3x,'|Bf3rd|')
c
        if(iprint.eq.1) WRITE(3,1)
c
c       WRITE(*,21) ns,Ggamma,ROTAD, Sxf, Syf, Szf, ANGX,ANGY,ANGZ, Spin
```

```
 21     FORMAT(3X,i5,F9.5,F10.3,3F10.4,
     1  3F10.3,F10.4,3F10.3,4F10.3)
c
c
        WRITE(3,21)
     1  ns,Ggamma,ROTAD,Sxf,Syf,Szf,
     1  ANGX,ANGY,ANGZ, Spin,Bf_1st,Bf_2nd,Bf_3rd
     1  ,ABS(Szf),ABS(Bf_1st),ABS(Bf_2nd),ABS(Bf_3rd)
c
        iprint=0
c
        return
        end
c**********************************************************************

c**********************************************************************
c
        subroutine Print_Res_at_Ex4
     1  (ns,Ggamma,Sxf,Syf,Szf,FRM_2X2,iprint,Bf_1st,Bf_2nd,Bf_3rd)
c
        COMPLEX*16 FRM_2X2(2,2)
        COMPLEX*16 TRACE
c
        REAL*8     pi, COSR, ACOSR, ROTAD,Sxf,Syf,Szf,Spin,Ggamma
        REAL*8     ANGX,ANGY,ANGZ
c
c       write(*,*) ' iprint in ex4' , iprint
c       read(*,*)
        pi=3.141592654
c
        TRACE=FRM_2X2(1,1)+FRM_2X2(2,2)
        COSR=0.5*DREAL(TRACE)
        ACOSR=2.0*DACOS(COSR)
        ROTAD=(180.0/PI)*(ACOSR)
c
        ANGX=(180.0/PI)*DACOS(Sxf)
        ANGy=(180.0/PI)*DACOS(Syf)
        ANGz=(180.0/PI)*DACOS(Szf)
c
           Spin=DSQRT(Sxf*Sxf+Syf*Syf+Szf*Szf)
c
c       if(iprint.eq.1) WRITE(*,1)
1       FORMAT('# Final results ',/,
     1  '#  ns     Ggamma   Rot-ANG',5X,'cosSx',5X,'cosSy'
     1  ,5x,'cosSz',5X,'ANGX',7X,'ANGY',5X,'ANGZ'
     1  ,6X,' MOD',3x,' B_fld_1st',1x,' B_fld_2nd',1x,' B_fld_3rd',3x,'|Szf|'
     1  ,2x,'|Bf1st|',2x,'|Bf2nd|',2x,'|Bf3rd|')
c
        if(iprint.eq.1) WRITE(4,1)
c       write(*,*) 'wrote in f4'
c
c       if(iprint.eq.1) WRITE(3,1)
c
c       WRITE(*,21) Ggamma,ROTAD, Sxf, Syf, Szf, ANGX,ANGY,ANGZ, Spin
 21     FORMAT(3X,i5,F9.5,F10.3,3F10.4,3F10.3,F10.4,3F10.3,F10.3,3F10.3)
c
```

```
c
        WRITE(4,21)
     1  ns,Ggamma,ROTAD,Sxf,Syf,Szf,
     1  ANGX,ANGY,ANGZ, Spin,Bf_1st,Bf_2nd,Bf_3rd
     1  ,ABS(Szf),ABS(Bf_1st),ABS(Bf_2nd),ABS(Bf_3rd)
c
        iprint=0
c       write(*,*) 'iprint 0',iprint
c
        return
        end
```