A Numerical Solution to the Equation that Relates the Torons to the Flux in 2D Non abelian Gauge Theory

Newton-Raphson Method

Dibakar Sigdel

Summer-2013

Contents

Part I

THEORY

1 INTRODUCTION

Let $W \in U(N)$ defines the total non-abelian flux operator on a two dimensional torous. Let $T, D \in SU(N)$ define the two toron operators on a two dimensional torous with D being diagonal. These three matrices satisfy the realtion $DTD^{\dagger}T^{\dagger} = W$. Assuming W is given we provide the details for the numerical computation of D and T by adapting the Newton-Raphson technique.

1.1 PROBLEM SET-UP

We want to solve for D and T in the equation

$$DTD^{\dagger}T^{\dagger} = W, \tag{1}$$

where W is given.
D is a diagonal matrix . We can write it as

$$D_{ij} = e^{i\phi_i}\delta ij. \tag{2}$$

We can rewrite equation (1) as

$$D^{\dagger}WT = TD^{\dagger} \tag{3}$$

For fixed W the product $D^{\dagger}W$ is a function of $\{\phi_i\}$.Let,

$$U(\phi) = D^{\dagger}W \tag{4}$$

such that, $U_{ij}(\phi) = e^{(-i\phi_i)}W_{ij}$

Let $e^k(\phi)$ and $\lambda_k(\phi)$ are eigenvector and eigenvalue of operator $U(\phi)$. Then we can set up an auxillary eigenvalue problem

$$U(\phi)e^k(\phi) = \lambda_k(\phi)e^k(\phi); \qquad k = 1, ..., N; \tag{5}$$

We can assume that the eigen vectors are normalized:

$$[e^k(\phi)]^\dagger e^k(\phi) = 1; \qquad k = 1, ..., N. \tag{6}$$

Now we prove the following:

- ***The eigen values*** $\lambda_k(\phi)$ ***of*** $U(\phi)$ ***are unimodular***.

i.e.

$$\lambda_k^*(\phi)\lambda_k(\phi) = 1, \qquad k = 1, ..., N \tag{7}$$

Proof:

From (5) and using (6) we get,

$$\lambda_k(\phi) = [e^k(\phi)]^\dagger U(\phi) e^k(\phi). \tag{8}$$

Since $U^\dagger U = UU^\dagger = I$, again from (5),

$$[e^k(\phi)]^\dagger = [e^k(\phi)]^\dagger U(\phi)\lambda_k^*(\phi). \tag{9}$$

Now,using thes two equations above,

$$\begin{aligned} \lambda_k(\phi)\lambda_k^*(\phi) &= \{[e^k(\phi)]^\dagger U(\phi) e^k(\phi)\}\lambda_k^*(\phi) \\ &= \{[e^k(\phi)]^\dagger U(\phi)\lambda_k^*(\phi)\} e^k(\phi) \\ &= [e^k(\phi)]^\dagger e^k(\phi) \\ &= 1 \blacksquare \end{aligned} \tag{10}$$

Using above facts (5) and (10), We can rewrite the above eigen value problem in 4 different ways:

$$\begin{aligned} U(\phi)e^k(\phi) &= \lambda_k(\phi)e^k(\phi); \\ U^\dagger(\phi)e^k(\phi) &= \lambda_k^*(\phi)e^k(\phi); \\ [e^k(\phi)]^\dagger U(\phi) &= \lambda_k(\phi)[e^k(\phi)]^\dagger; \\ [e^k(\phi)]^\dagger U^\dagger(\phi) &= \lambda_k^*(\phi)[e^k(\phi)]^\dagger; \\ k &= 1, ..., N. \end{aligned} \tag{11}$$

Furthermore, due to the unimodular property of eigenvalue we can write ;

$$\lambda_k = e^{-i\theta_k(\phi)} \tag{12}$$

- ***The eigen vectors*** $e^k(\phi)$ ***of*** $U(\phi)$ ***are orthonormal***.

i.e.

$$[e^j(\phi)]^\dagger e^k(\phi) = \delta jk \tag{13}$$

Proof:
Multiplying the first of the four equations in (11) by $e^j(\phi)$ from left,

$$[e^j(\phi)]^\dagger U(\phi) e^k(\phi) = e^{-i\theta_k(\phi)}[e^j(\phi)]^\dagger e^k(\phi). \tag{14}$$

Setting k = j in the third of the four equations in (11) and then multiplying by $e^k(\phi)$ from right results

$$[e^j(\phi)]^\dagger U(\phi) e^k(\phi) = e^{-i\theta_j(\phi)}[e^j(\phi)]^\dagger e^k(\phi). \tag{15}$$

From above two equation:

$$e^{-i\theta_j(\phi)} - e^{-i\theta_k(\phi)}[e^j(\phi)]^\dagger e^k(\phi) = 0. \tag{16}$$

If $j \neq k$ and $\theta_j \neq \theta_k$, along with normalization condition (6) this results in the statement (13)■

Let us define a matrix of eigen vectors,

$$V_{jk} = e^k_j(\phi) \tag{17}$$

Using equation (13),we get $V^\dagger(\phi)V(\phi) = I$.

Note: *The normalization condition in* (6) *does not fix the phase of the eigenvectors,* $e^k(\phi)$, *but we will not need to make any phase choice for our calculation.*

We would have found the solution to (1), if

$$\theta_k(\phi) = \phi_k; \qquad k = 1, ..., N. \tag{18}$$

Furthermore the corresponding $V(\phi)$ can be used to set

$$T = \frac{V}{[detV]^{\frac{1}{N}}} \tag{19}$$

Then the eigen value equation would exactly coincides with

$$D^{\dagger}WT = TD^{\dagger}. \tag{20}$$

In order to obtain the numerical solution to (1),we combine the auxillary eigenvalue equation (5), with action

$$S(\phi) = \sum_{i=1}^{N} \sin^2[\frac{1}{2}(\theta_i - \phi_i)] \tag{21}$$

If we have obtained a minimum of the action $S(\phi)$, then we have found a solution to (1). We will assume the W is such that $\phi_i \neq \phi_j$ if $i \neq j$.

2 NEWTON-RAPHSON METHOD

We will use the Newton-Raphson method to find the minimum of (21). Let $\phi^{(n)}$ be one choice of ϕ. We write the expansion of $S(\phi)$ around $\phi^{(n)}$ up to the quadratic term as

$$S(\phi) = S(\phi^{(n)}) + \delta\phi^{(n)}.\nabla S\left(\phi^{(n)}\right) + \frac{1}{2}\delta\phi^{(n)}.\mathcal{H}S\left(\phi^{(n)}\right).\delta\phi^{(n)} + ... \tag{22}$$

Where,

$$\begin{aligned} [\delta\phi^{(n)}]_l = \phi_l - \phi_l^{(n)}; \\ [\nabla S\left(\phi^{(n)}\right)]_l = \left.\frac{\partial S(\phi)}{\partial \phi_l}\right|_{\phi^{(n)}}; \\ [\mathcal{H}S\left(\phi^{(n)}\right)]_{lk} = [\mathcal{H}S\left(\phi^{(n)}\right)]_{kl} = \left.\frac{\partial^2 S(\phi)}{\partial \phi_k \partial \phi_l}\right|_{\phi^{(n)}}. \end{aligned} \tag{23}$$

We choose $\phi^{(n)}$ as the point where the action up to the quadratic term reaches an extremum and this is given by

$$\delta\phi_l = \phi_l^{(n+1)} - \phi_l^{(n)} = [\mathcal{H}S\left(\phi^{(n)}\right)]^{-1}.\nabla S\left(\phi^{(n)}\right)]_l. \tag{24}$$

This iterative procedure will terminate at the extremum given by

$$\nabla S(\phi^{extremum}) = 0. \tag{25}$$

2.1 Cartan's method of transformation

Since matrix $[\mathcal{H}S\left(\phi^{(n)}\right)]$ is a symmetric matrix, it is also a singular matrix and hence the inverse of this matrix does not exist as mentioned in (24). To make the iteration procedure possible, we do the Cartan's method of transformation so that matrix $[\mathcal{H}S\left(\phi^{(n)}\right)]$ becomes nonsingular.

We consider the transformation as the projection defined by

$$\overline{\delta\phi} = \mathcal{M}\delta\phi \tag{26}$$

Where $\overline{\delta\phi}$ is the projection of vector $\delta\phi$ from N to $N-1$ dimensional vector space. For this, we take a transformation matrix $\mathcal{M}$ as

$$\begin{aligned}
(\mathcal{M})_{pq} &= \tfrac{1}{\sqrt{p(p+1)}}, && \text{if } (p<N, q\le p)\\
&= -\tfrac{(p+1)}{\sqrt{p(p+1)}}, && \text{if } (p<N, q=p+1)\\
&= 0, && \text{if } (p<N, q>p+1)\\
&= \tfrac{1}{\sqrt{N}}, && \text{if } (p=N)
\end{aligned}$$

i.e.

$$\mathcal{M}_{(3\times3)} = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & \frac{-2}{\sqrt{6}} \\ \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \end{pmatrix} \tag{27}$$

$$\mathcal{M}_{(5\times5)} = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0 & 0 & 0 \\ \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & \frac{-2}{\sqrt{6}} & 0 & 0 \\ \frac{1}{\sqrt{12}} & \frac{1}{\sqrt{12}} & \frac{1}{\sqrt{12}} & \frac{-3}{\sqrt{12}} & 0 \\ \frac{1}{\sqrt{20}} & \frac{1}{\sqrt{20}} & \frac{1}{\sqrt{20}} & \frac{1}{\sqrt{20}} & \frac{-4}{\sqrt{20}} \\ \frac{1}{\sqrt{5}} & \frac{1}{\sqrt{5}} & \frac{1}{\sqrt{5}} & \frac{1}{\sqrt{5}} & \frac{1}{\sqrt{5}} \end{pmatrix} \tag{28}$$

From (26), plugging $\delta\phi = \mathcal{M}^T\overline{\delta\phi}$ in (22), we find,

$$\begin{aligned}
\overline{S}(\overline{\phi}) &= \overline{S}(\overline{\phi^{(n)}}) + [\mathcal{M}^T\overline{\delta\phi^{(n)}}]^T.\nabla S\left(\phi^{(n)}\right) + \frac{1}{2}[\mathcal{M}^T\overline{\delta\phi^{(n)}}]^T.\mathcal{H}S\left(\phi^{(n)}\right).\mathcal{M}^T\overline{\delta\phi^{(n)}} + ...\\
&= \overline{S}(\overline{\phi^{(n)}}) + [\overline{\delta\phi^{(n)}}]^T.[\mathcal{M}.\nabla S\left(\phi^{(n)}\right)] + \frac{1}{2}[\overline{\delta\phi^{(n)}}]^T.[\mathcal{M}.\mathcal{H}S\left(\phi^{(n)}\right).\mathcal{M}^T].\overline{\delta\phi^{(n)}} + ... \quad (29)
\end{aligned}$$

Let,

$$\nabla\overline{S}\left(\overline{\phi^{(n)}}\right) = \mathcal{M}.\nabla S\left(\phi^{(n)}\right)$$
$$\overline{\mathcal{H}S}\left(\overline{\phi^{(n)}}\right) = \mathcal{M}.\mathcal{H}S\left(\phi^{(n)}\right).\mathcal{M}^T \tag{30}$$

Then (22) in projected space looks like

$$\overline{S}(\overline{\phi}) = \overline{S}(\overline{\phi^{(n)}}) + \overline{\delta\phi^{(n)}}.\nabla\overline{S}\left(\overline{\phi^{(n)}}\right) + \frac{1}{2}\overline{\delta\phi^{(n)}}.\overline{\mathcal{H}S}\left(\overline{\phi^{(n)}}\right).\overline{\delta\phi^{(n)}} + ... \tag{31}$$

Clearly our equation (24) in projected space looks like

$$\overline{\delta\phi_l} = [\overline{\mathcal{H}S}\left(\overline{\phi^{(n)}}\right)]^{-1}.[\overline{\nabla S}\left(\overline{\phi^{(n)}}\right)]_l. \tag{32}$$

This $\overline{\delta\phi_l}$ is transformed back to the original N dimensional space,

$$\delta\phi = \mathcal{M}^T.\overline{\delta\phi}. \tag{33}$$

This completes one cycle of our iteration process and we get new set of ϕ,

$$\phi^{n+1} = \delta\phi + \phi^n. \tag{34}$$

2.2 Matix elements of $\mathcal{H}S$ and ∇S

Action is

$$S(\phi) = \sum_{i=1}^{N} \sin^2[\frac{1}{2}(\theta_i - \phi_i)] \tag{35}$$

Let ,

$$s_i(\phi) = \sin^2[\frac{1}{2}(\theta_i - \phi_i)] \tag{36}$$

First derivative of s_i with respect to ϕ_l is given by,

$$\frac{\partial s_i(\phi)}{\partial\phi_l} = s_i(\phi)c_i(\phi)\left(\frac{\partial\theta_i(\phi)}{\partial\phi_l} - \delta_{il}\right)$$
$$= s_i(\phi)c_i(\phi)\frac{\partial\theta_i(\phi)}{\partial\phi_l} - s_i(\phi)c_i(\phi)\delta_{il} \tag{37}$$

Where,

$$c_i = \cot[\frac{1}{2}(\theta_i(\phi) - \phi_i)] \tag{38}$$

Again differentiating $\frac{\partial s_i(\phi)}{\partial \phi_l}$ with respect to ϕ_m,

$$\frac{\partial^2 s_i(\phi)}{\partial \phi_l \partial \phi_m} = \frac{\partial s_i(\phi)}{\partial \phi_m} c_i(\phi) \frac{\partial \theta_i(\phi)}{\partial \phi_l} + s_i(\phi) \frac{\partial c_i(\phi)}{\partial \phi_m} \frac{\partial \theta_i(\phi)}{\partial \phi_l} + s_i(\phi) c_i(\phi) \frac{\partial^2 \theta_i(\phi)}{\partial \phi_l \partial \phi_m} - \frac{\partial s_i(\phi)}{\partial \phi_m} c_i(\phi) \delta_{il} - s_i(\phi) \frac{\partial c_i(\phi)}{\partial \phi_m} \delta_{il}. \tag{39}$$

Now

$$[\mathcal{H}S]_{lm} = \sum_{i=1}^{N} \frac{\partial^2 s_i(\phi)}{\partial \phi_l \partial \phi_m}$$
$$[\nabla S]_l = \sum_{i=1}^{N} \frac{\partial s_i(\phi)}{\partial \phi_l} \tag{40}$$

3 PERTURBATION THEORY

We set up second order perturbation theory for the computation of $\frac{\partial \theta_i(\phi)}{\partial \phi_l}$ and $\frac{\partial^2 \theta_i(\phi)}{\partial \phi_l \partial \phi_m}$ at $\phi^{(n)}$.We will use the following short hand notations:

$$U = U(\phi^{(n)}); \qquad U'_l = \frac{\partial U(\phi^{(n)})}{\partial \phi_l}; \qquad U''_{lm} = U''_{ml} = \frac{\partial^2 U(\phi^{(n)})}{\partial \phi_m \partial \phi_l} \tag{41}$$

$$e^{(k)} = e^{(k)}(\phi^{(n)}); \qquad e_l^{(k)'} = \frac{\partial e^{(k)}(\phi^{(n)})}{\partial \phi_l}; \qquad e_{lm}^{(k)''} = e_{ml}^{(k)''} = \frac{\partial^2 e^{(k)}(\phi^{(n)})}{\partial \phi_m \partial \phi_l} \tag{42}$$

$$\lambda_k = \lambda_k(\phi^{(n)}); \qquad \lambda'_{kl} = \frac{\partial \lambda_k(\phi^{(n)})}{\partial \phi_l}; \qquad \lambda''_{klm} = \lambda''_{kml} = \frac{\partial^2 \lambda_k(\phi^{(n)})}{\partial \phi_m \partial \phi_l} \tag{43}$$

$$\delta_l = \phi_l - \phi_l^{(n)} \tag{44}$$

Using this notation, we can write (5) as

$$\left[U + \sum_l \delta_l U'_l + \frac{1}{2} \sum_{lm} \delta_l \delta_m U''_{lm} + ...\right] \left[e^{(k)} + \sum_l \delta_l e_l^{(k)'} + \frac{1}{2} \sum_{lm} \delta_l \delta_m e_{lm}^{(k)''} + ...\right]$$

$$= \left[\lambda_k + \sum_l \delta_l \lambda'_{kl} + \frac{1}{2}\sum_{lm} \delta_l \delta_m \lambda''_{klm} + ...\right]\left[e^{(k)} + \sum_l \delta_l e_l^{(k)\prime} + \frac{1}{2}\sum_{lm}\delta_l\delta_m e_{lm}^{(k)\prime\prime} + ...\right] \quad (45)$$

We can write condition of orthonormality, (6),as

$$\left[e^{(j)} + \sum_l \delta_l e_l^{(j)\prime} + \frac{1}{2}\sum_{lm}\delta_l\delta_m e_{lm}^{(j)\prime\prime} + ...\right]^\dagger \left[e^{(k)} + \sum_l \delta_l e_l^{(k)\prime} + \frac{1}{2}\sum_{lm}\delta_l\delta_m e_{lm}^{(k)\prime\prime} + ...\right] = \delta_{jk} \quad (46)$$

Since $U(\phi)$ has to be unitary for all ϕ, we have

$$\left[U + \sum_l \delta_l U'_l + \frac{1}{2}\sum_{lm}\delta_l\delta_m U''_{lm} + ...\right]^\dagger \left[U + \sum_l \delta_l U'_l + \frac{1}{2}\sum_{lm}\delta_l\delta_m U''_{lm} + ...\right] = 1 \quad (47)$$

At zeroth order

At the lowest order we have

$$Ue^{(k)} = \lambda_k e^{(k)}; \qquad e^{(j)\dagger}e^{(k)} = \delta_{jk} \quad (48)$$

Given ϕ^n, we would form U as in (5). In order to find all $e^{(k)}$,we would form the hermitian matrix

$$H = \frac{1}{2}(U + U^\dagger). \quad (49)$$

Assuming no degeneracies, the eigenvectors of U and H are the same. One can use a standard numerical algorithm to compute all eigenvectors, $e^{(k)}$ of H. Once we have computed the eigenvectors, we can use

$$\lambda_k = [e^{(k)}]^\dagger U e^{(k)} \quad (50)$$

to compute the eigenvalues.

At first order

At first order, we need the term multiplying δ_l to be zero for all l. From (45), we get

$$Ue_l^{(k)\prime} + U'_l e^{(k)} = \lambda_k e_l^{(k)\prime} + \lambda'_{kl} e^{(k)} \quad (51)$$

From (46), we get

$$[e_l^{(j)'}]^\dagger e^{(k)} + e^{(j)\dagger} e_l^{(k)'} = 0. \tag{52}$$

From unitarity condition (47),we get

$$U_l^{'\dagger} U + U^\dagger U_l^{'} = 0. \tag{53}$$

We can use orthonormal set $e^{(k)}$, to write

$$e_l^{(k)'} = \sum_p C_{kp}^l e^{(p)}. \tag{54}$$

Inserting (54) in (52) results in

$$(C_{jk}^l)^* + (C_{kj}^l) = 0 \tag{55}$$

We insert (54) in (51) and take the inner product with $e^{q\dagger}$. Using (48), we arive at

$$e^{(q)\dagger} U_l^{'} e^{(k)} = (\lambda_k - \lambda_q) C_{kq}^l + \lambda_{kl}^{'} \delta_{kq}. \tag{56}$$

seeting $k = q$ we get,

$$\lambda_{kl}^{'} = e^{(k)\dagger} U_l^{'} e^{(k)} \tag{57}$$

For $q \neq k$, we find

$$C_{kq}^l = \frac{e^{(q)\dagger} U_l^{'} e^{(k)}}{(\lambda_k - \lambda_q)} \tag{58}$$

- ***Above equation of cofficient is consistent with*** (55).

Proof: We can use (53) to write

$$U_l^{'\dagger} = -U^\dagger U'_l U^\dagger. \tag{59}$$

Therefore (58), can be written using the above equation as

$$\begin{aligned}(C_{jk}^l)^* &= \frac{e^{(j)\dagger} U_l^{\dagger'} e^{(k)}}{(\lambda_j^* - \lambda_k^*)} = -\frac{e^{(j)\dagger} U^\dagger U'_l U^\dagger e^{(k)}}{(\lambda_j^* - \lambda_k^*)} \\ &= -\frac{\lambda_k^* \lambda_j^* e^{(j)\dagger} U_l^{'} e^{(k)}}{(\lambda_j^* - \lambda_k^*)} = -\frac{e^{(j)\dagger} U_l^{'} e^{(k)}}{(\frac{1}{\lambda_k^*} - \frac{1}{\lambda_j^*})}\end{aligned}$$

$$= -\frac{e^{(j)\dagger}U_l' e^{(k)}}{(\lambda_k - \lambda_j)}$$
$$= -C_{kj}^l. \tag{60}$$

The first equality in the second line comes from (11) and the statement, (7). The second last equality follows from the statement, (7).Finally, the last term comes from (58) and we have proven the consistency with (55) for $k \neq j$. When $k = j$, the condition on (55) says that

$$(C_{kk}^l)^* + (C_{kk}^l) = 0 \tag{61}$$

Note:*The imaginary part of C_{kk}^l can not be fixed due to the freedom in phase choice of eigenvectors*■

At second order

At second order,We need the term multiplying $\delta_l \delta_m$ to be zero for all l and m. From (45), we get

$$U_{lm}^{''} e^{(k)} + U_l^{'} e_m^{(k)'} + U_m^{'} e_l^{(k)'} + U e_{lm}^{(k)''} = \lambda_{klm}^{''} e^{(k)} + \lambda_{kl}^{'} e_m^{(k)'} + \lambda_{km}^{'} e_l^{(k)'} + \lambda_k e_{lm}^{(k)''} \tag{62}$$

We take inner product with $e^{(k)\dagger}$ and use use (11) and (54) to get

$$e^{(k)\dagger} U_{lm}^{''} e^{(k)} + \sum_p e^{(k)\dagger} U_l^{'} e^{(p)} C_{kp}^m + \sum_p e^{(k)\dagger} U_m^{'} e^{(p)} C_{kp}^l + \lambda_k e^{(k)\dagger} e_{lm}^{(k)''}$$
$$= \lambda_{klm}^{''} + \lambda_{kl}^{'} C_{kk}^m + \lambda_{km}^{'} C_{kk}^l + \lambda_k e^{(k)\dagger} e_{lm}^{(k)''} \tag{63}$$

The last term on both side cancel out. Using (57), we see that the $p = k$ term in the second and third term on the left cancels out the second and third term on the right. Next , we use (58) to write out the $p \neq k$ terms in the sum on the left . These operations results in

$$\lambda_{klm}^{''} = e^{(k)\dagger} U_{lm}^{''} e^{(k)} + \sum_{p \neq k} [C_{pk}^l C_{kp}^m + C_{pk}^m C_{kp}^l](\lambda_p - \lambda_k) \tag{64}$$

3.1 Matrix Elements of $U^{'}$ and $U^{''}$

We need to compute explicit expressions for the matrix elemnt that appear in (57) and (64) . Given that

$$U_{ij} = e^{-i\phi_i} W_{ij} \tag{65}$$

We can obtain the following quantities in the equation (41):

$$[U_l^{'}]_{ij} = -i\delta_{il} U_{ij}; \qquad [U_{lm}^{''}]_{ij} = -i\delta_{il}\delta_{im} U_{ij} \tag{66}$$

3.2 Relation Between Derivatives of λ and θ_k

Since

$$\lambda_k = e^{-i\theta_k}, \tag{67}$$

We have

$$\frac{\partial\theta_k(\phi^{(n)})}{\partial\phi_l} = i\lambda_k^* \lambda_{kl}^{'} \tag{68}$$

and

$$\frac{\partial^2\theta_k(\phi^{(n)})}{\partial\phi_l \partial\phi_m} = -i(\lambda_k^*)^2 \lambda_{km}^{'} + i\lambda_k^* \lambda_{klm}^{''} \tag{69}$$

Note: *These quanties are real which has been proved at the Appendix.*

4 ITERATION PROCESS

To begin the iteration process, we start with a randomly generated W matrix which is fixed for the whole iteration. Next we take a set of (ϕ_k) as a first input set where, $\sum_k \phi_k = 0$. We emerge with a new set of (ϕ_k) after the first cycle, which becomes new input set of (ϕ_k) in second cycle of iteration. Following are some important steps of the iteration process:

- ***Generating SU(N) matrix***.

We can randomly generate $SU(N)$ matrix of any N with the help of randomly generated $SU(2)$ matix.

Let α, β and γ defined in the domain; $-\pi < \alpha, \beta < \pi$ and $-\frac{\pi}{2} < \gamma < \frac{\pi}{2}$. If r_1, r_2, r_3 are random numbers in between 1 and 0, then we can generate random angles α, β, γ defined in those specific domain.

$$\alpha = \pi(2r_1 - 1); \qquad \beta = \pi(2r_2 - 1); \qquad \gamma = \frac{\pi}{2}(2r_3 - 1) \tag{70}$$

$$SU(2) = \begin{pmatrix} \cos(\gamma)e^{i\alpha} & \sin(\gamma)e^{i\beta} \\ -\sin(\gamma)e^{-i\beta} & \cos(\gamma)e^{-i\alpha} \end{pmatrix} = \begin{pmatrix} a & b \\ -b^* & a* \end{pmatrix} \tag{71}$$

Once we construct the $SU(2)$ matrix elements, we can supply these 4 matrix elements to construct a certain number of SU(N) matrices by suitable choice of permutations. The product of all those SU(N) matrices formed by permutations is a required SU(N) matrix.

For any N, First SU(N) matrix is formed by selecting two diagonal-position where we supply SU2(1,1) and SU2(2,2) and remaining diagonal elements are equal to 1. We draw two horizontal and verticle lines intersecting at those diogonal position. Other elements SU2(1,2) and SU2(2,1) is supplied at the other off-diagonal intersecting position and rest of the position are filled with 0 as shown below.

$$SU(N) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & ... \\ 0 & 1 & 0 & 0 & 0 & 0 & ... \\ 0 & 0 & a & 0 & b & 0 & ... \\ 0 & 0 & 0 & 1 & 0 & 0 & ... \\ 0 & 0 & -b^* & 0 & a^* & 0 & ... \\ 0 & 0 & 0 & 0 & 0 & 1 & ... \\ . & . & . & . & . & . & ... \\ . & . & . & . & . & . & ... \\ . & . & . & . & . & . & ... \end{pmatrix} \tag{72}$$

We can repeat this process and can construct in general $\frac{N(N-1)}{2}$ independent SU(N) matrices and take the product of all those matrices.

For example:

For SU(3), It is a product of three SU(3) matrices formed by suitable permutation of those SU(2) elements:

$$SU(3) = \begin{pmatrix} a & b & 0 \\ -b^* & a^* & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a & 0 & b \\ 0 & 1 & 0 \\ -b^* & 0 & a^* \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & a & b \\ 0 & -b^* & a^* \end{pmatrix} \tag{73}$$

Hence in general for SU(N) matrix we can form $\frac{N(N-1)}{2}$ independent SU(N) matrices by using SU(2) elements. The required SU(N) is the product of all those independent matrices.

- ***Rearraning eigen vectors and eigen values of matrix U in every cycle***.

Once we get the eigen vector of matrix U by constructing Hermitian matrix H as in equation (49), we can calculate eigen values of U by using (50).

From equation (12),

$$\lambda_k = e^{-i\theta_k(\phi)} \tag{74}$$

We can calculate,

$$\theta_k = i \log_e \lambda_k \tag{75}$$

To make sure that these θ_k are within the domain$[-\pi, \pi]$, We take ,

$$\theta_k = -i \log_e e^{-i\theta_k} \tag{76}$$

Since we can get $\sum_k \theta_k = 0$, similar to input set ϕ_k, we can think both set to lie on a plane passing through the origin. After getting a set of θ_k, we can rearrange them in different permutations. We will select that particular permutation θ_{p_k} which lies closer to the set of ϕ_k and lead to minimise the action in subsequent cycle. Our aim in every cycle of iteration is to reach with value of θ_k closer and closer to the value of ϕ_k and ultimately iteration stops at $\theta_k = \phi_k$, where S = 0.To do this we measure the distance between set of ϕ_k and θ_{p_k} by

$$\Delta_p(\theta, \phi) = \sqrt{\sum_k^N (\theta_{p_k} - \phi_k)^2}, \tag{77}$$

and we take the minimum from the set of all distances $\Delta_p(\theta, \phi)$. This will give us a perticular order of θ_k in a new set θ_{p_k}. We use this set of θ_{p_k} to rearrange both eigenvaluses and eigenvectors of U matrix.

Note: *Rearranging the eigenvalues and eigen vectors of matrix U does not affect the whole theory. As the matter of fact, this rearrangement is a rotation of the eigen vectors from old set θ_k to the new set θ_{p_k}*

5 APPENDIX

5.1 To show $[\theta'_k]_l$ is a real quantity

We have :

$$\begin{aligned} [\theta'_k]_l &= i\lambda_k^* \lambda'_{kl} \\ \therefore [\theta'_k]_l^* &= -i\lambda_k \lambda'^*_{kl} \end{aligned} \tag{78}$$

We also have,$\lambda'_{kl} = e^{(k)\dagger} U'_l e^{(k)}$

$$[\theta'_k]_l^* = -i\lambda_k e^{(k)\dagger} U'^{\dagger}_l e^{(k)} \tag{79}$$

Now using $U'^{\dagger}_l = -U^\dagger U'_l U^\dagger$

$$\begin{aligned} [\theta'_k]_l^* &= -i\lambda_k e^{(k)\dagger}[-U^\dagger U'_l U^\dagger] e^{(k)} \\ [\theta'_k]_l^* &= i\lambda_k (e^{(k)\dagger} U^\dagger) U'_l (U^\dagger e^{(k)}) \\ [\theta'_k]_l^* &= i\lambda_k [\lambda_k^*]^2 [e^{(k)\dagger} U'_l e^{(k)}] \\ [\theta'_k]_l^* &= i[\lambda_k^*][e^{(k)\dagger} U'_l e^{(k)}] \\ [\theta'_k]_l^* &= i[\lambda_k^*]\lambda'_{kl} \\ \implies [\theta'_k]_l^* &= [\theta'_k]_l \end{aligned} \tag{80}$$

This implies $[\theta'_k]_l$ is real quantity.

5.2 To show$[\theta''_k]_{lm}$ is a real quantity

We have :

$$
\begin{gathered}
[\theta''_k]_{lm} = -i(\lambda_k^*)^2\lambda'_{km}\lambda'_{kl} + i\lambda_k^*\lambda''_{klm} \\
\therefore [\theta''_k]^*_{lm} = i(\lambda_k)^2\lambda'^*_{km}\lambda'^*_{kl} - i\lambda_k\lambda''^*_{klm}
\end{gathered}
\tag{81}
$$

- Consider the first term in equation (81):$i(\lambda_k)^2\lambda'^*_{km}\lambda'^*_{kl}$,
  using $\lambda'^*_{km} = -(\lambda_k^*)^2\lambda'_{km}$ as in deriving equation 80 above,we get

$$
i(\lambda_k)^2\lambda'^*_{km}\lambda'^*_{kl} = i(\lambda_k)^2[\lambda'_{km}(\lambda_k^*)^2][\lambda'_{kl}(\lambda_k^*)^2] = i(\lambda_k^*)^2\lambda'_{km}\lambda'_{kl} \tag{82}
$$

- Consider the second term in equation (81) :$-i\lambda_k\lambda''^*_{klm}$
  Where $\lambda''_{klm} = e^{(k)\dagger}U''_{lm}e^{(k)} - \sum_{p\neq k}[c^l_{pk}c^m_{kp} + c^m_{pk}c^l_{kp}](\lambda_k - \lambda_p)$

$$
\therefore -i\lambda_k\lambda''^*_{klm} = -i\lambda_k(e^{(k)\dagger}U''_{lm}e^{(k)})^\dagger + i\lambda_k\sum_{p\neq k}[c^{l*}_{pk}c^{m*}_{kp} + c^{m*}_{pk}c^{l*}_{kp}](\lambda_k^* - \lambda_p^*) \tag{83}
$$

- Consider the first term in equation (83):$i\lambda_k(e^{(k)\dagger}U''_{lm}e^{(k)})^\dagger$,
  using $U''^\dagger_l = -U^\dagger U''_{lm}U^\dagger - U'^\dagger_l U'_m U^\dagger - U'^\dagger_m U'_l U^\dagger$,we get

$$
\begin{aligned}
-i\lambda_k(e^{(k)\dagger}U''_l e^{(k)})^\dagger &= -i\lambda_k(e^{(k)\dagger}U''^\dagger_{lm}e^{(k)}) \\
&= i\lambda_k(e^{(k)\dagger}[U^\dagger U''_{lm}U^\dagger + U'^\dagger_l U'_m U^\dagger + U'^\dagger_m U'_l U^\dagger]e^{(k)}) \\
&= i\lambda_k[(e^{(k)\dagger}U^\dagger U''_{lm}U^\dagger e^{(k)}) + (e^{(k)\dagger}U'^\dagger_l U'_m U^\dagger e^{(k)}) + (e^{(k)\dagger}U'^\dagger_m U'_l U^\dagger e^{(k)})] \\
&= i\lambda_k[(\lambda_k^*)^2(e^{(k)}U''_{lm}e^{(k)}) + \lambda_k^*(e^{(k)\dagger}U'^\dagger_l U'_m e^{(k)}) + \lambda_k^*(e^{(k)\dagger}U'^\dagger_m U'_l e^{(k)})]
\end{aligned}
\tag{84}
$$

- Consider the term:$e^{(k)\dagger}U_l'^{\dagger}U_m'e^{(k)}$ and using $\sum e^{(p)}e^{(p)\dagger}=I$

$$\begin{aligned}
e^{(k)\dagger}U_l'^{\dagger}U_m'e^{(k)} &= \sum_p e^{(k)\dagger}U_l'^{\dagger}e^{(p)}e^{(p)\dagger}U_m'e^{(k)} \\
= e^{(k)\dagger}U_l'^{\dagger}e^{(k)}e^{(k)\dagger}U_m'e^{(k)} &+ \sum_{k\neq p} e^{(k)\dagger}U_l'^{\dagger}e^{(p)}e^{(p)\dagger}U_m'e^{(k)} \\
&= \lambda_{kl}'^{*}\lambda_{km}' + \sum_{k\neq p}(e^{(p)\dagger}U_l'e^{(k)})^{\dagger}(e^{(p)\dagger}U_m'e^{(k)}) \\
&= \lambda_{kl}'^{*}\lambda_{km}' + \sum_{k\neq p} C_{kp}^{l*}(\lambda_k^*-\lambda_p^*)C_{kp}^{m}(\lambda_k-\lambda_p) \qquad (85)
\end{aligned}$$

Now we use the fact that : $C_{kp}^{l*}=-C_{pk}^{l}$
and $(\lambda_k^*-\lambda_p^*)=\lambda_k^*\lambda_p^*(\lambda_p-\lambda_k)=-\lambda_k^*\lambda_p^*(\lambda_k-\lambda_p)$
and $\lambda_{kl}'^{*}=-(\lambda_k^*)^2\lambda_{kl}'$

$$\therefore e^{(k)\dagger}U_l'^{\dagger}U_m'e^{(k)} = -(\lambda_k^*)^2\lambda_{kl}'\lambda_{km}' + \lambda_k^*\sum_{k\neq p}\lambda_p^*(\lambda_k-\lambda_p)^2C_{pk}^{l}C_{kp}^{m} \qquad (86)$$

Similarly we get:

$$e^{(k)\dagger}U_m'^{\dagger}U_l'e^{(k)} = -(\lambda_k^*)^2\lambda_{km}'\lambda_{kl}' + \lambda_k^*\sum_{k\neq p}\lambda_p^*(\lambda_k-\lambda_p)^2C_{pk}^{m}C_{kp}^{l} \qquad (87)$$

Now adding these two equations viz: (86) and (87),the second and third term of equation (84) becomes,

$$\begin{aligned}
\lambda_k^*[e^{(k)\dagger}U_l'^{\dagger}U_m'e^{(k)} + e^{(k)\dagger}U_m'^{\dagger}U_l'e^{(k)}] &= -2(\lambda_k^*)^3(\lambda_{kl}'\lambda_{km}') \\
+(\lambda_k^*)^2\sum_{k\neq p}\lambda_p^*(\lambda_k-\lambda_p)^2&(C_{pk}^{l}C_{kp}^{m}+C_{pk}^{m}C_{kp}^{l}) \qquad (88)
\end{aligned}$$

Pluging the value of equation (88) in equation (84),the first term of equation (83) becomes:

$$
\begin{aligned}
-i\lambda_k(e^{(k)\dagger}U''_{lm}e^{(k)})^\dagger &= i\lambda_k[(\lambda_k^*)^2(e^{(k)\dagger}U''_{lm}e^{(k)}) - 2(\lambda_k^*)^3(\lambda'_{kl}\lambda'_{km})] \\
&\quad +(\lambda_k^*)^2\sum_{k\neq p}\lambda_p^*(\lambda_k-\lambda_p)^2(C^l_{pk}C^m_{kp}+C^m_{pk}C^l_{kp}) \\
&= i\lambda_k(\lambda_k^*)^2[(e^{(k)\dagger}U''_{lm}e^{(k)}) - 2(\lambda_k^*)\lambda'_{kl}\lambda'_{km} \\
&\quad +\sum_{p\neq k}\lambda_p^*(\lambda_k-\lambda_p)^2(C^m_{pk}C^l_{kp}+C^m_{pk}C^l_{kp})] \\
&= i\lambda_k^*[(e^{(k)\dagger}U''_{lm}e^{(k)}) - 2(\lambda_k^*)\lambda'_{kl}\lambda'_{km} \\
&\quad +\sum_{p\neq k}\lambda_p^*(\lambda_k-\lambda_p)^2(C^m_{pk}C^l_{kp}+C^m_{pk}C^l_{kp})] \\
&= i(\lambda_k^*)(e^{(k)\dagger}U''_{lm}e^{(k)}) - 2i(\lambda_k^*)^2\lambda'_{kl}\lambda'_{km} \\
&\quad +i(\lambda_k^*)\sum_{p\neq k}\lambda_p^*\lambda_k(\lambda_k-\lambda_p)(C^m_{pk}C^l_{kp}+C^m_{pk}C^l_{kp})] \\
&\quad -i(\lambda_k^*)\sum_{p\neq k}(\lambda_k-\lambda_p)(C^m_{pk}C^l_{kp}+C^m_{pk}C^l_{kp})] \\
&= i(\lambda_k^*)(e^{(k)\dagger}U''_{lm}e^{(k)}) - 2i(\lambda_k^*)^2\lambda'_{kl}\lambda'_{km} \\
&\quad +i\sum_{p\neq k}\lambda_p^*(\lambda_k-\lambda_p)(C^m_{pk}C^l_{kp}+C^m_{pk}C^l_{kp})] \\
&\quad -i(\lambda_k^*)\sum_{p\neq k}(\lambda_k-\lambda_p)(C^m_{pk}C^l_{kp}+C^m_{pk}C^l_{kp})] \qquad (89)
\end{aligned}
$$

The second term in equation (83) becomes:

$$
\begin{aligned}
i\lambda_k\sum_{p\neq k}[c^{l*}_{pk}c^{m*}_{kp}+c^{m*}_{pk}c^{l*}_{kp}](\lambda_k^*-\lambda_p^*)] &= i\lambda_k\sum_{p\neq k}[c^l_{kp}c^m_{pk}+c^m_{kp}c^l_{pk}]\lambda_k^*\lambda_p^*(\lambda_p-\lambda_k)] \\
&= -i\sum_{p\neq k}\lambda_p^*[c^l_{kp}c^m_{pk}+c^m_{kp}c^l_{pk}](\lambda_k-\lambda_p)](90)
\end{aligned}
$$

When we plug the values from equation (89) and equation (90) to (83), we find that the third term in equation (89) is cancelled by the term we got in equation (90)
Hence,from (83) we have:

$$
\begin{aligned}
-i\lambda_k\lambda''^*_{klm} &= i(\lambda_k^*)(e^{(k)\dagger}U''_{lm}e^{(k)}) - 2i(\lambda_k^*)^2\lambda'_{kl}\lambda'_{km} - i(\lambda_k^*)\sum_{p\neq k}(\lambda_k-\lambda_p)(C^m_{pk}C^l_{kp}+C^m_{pk}C^l_{kp}) \\
-i\lambda_k\lambda''^*_{klm} &= i(\lambda_k^*)[(e^{(k)\dagger}U''_{lm}e^{(k)}) - \sum_{p\neq k}(\lambda_k-\lambda_p)(C^m_{pk}C^l_{kp}+C^m_{pk}C^l_{kp})] - 2i(\lambda_k^*)^2\lambda'_{kl}\lambda'_{km} \\
-i\lambda_k\lambda''^*_{klm} &= i\lambda_k^*\lambda''^*_{klm} - 2i(\lambda_k^*)^2\lambda'_{kl}\lambda'_{km}
\end{aligned}
\tag{91}
$$

Now we go back to equation (81) plugging the values from (91) and (82) we get

$$
\begin{aligned}
[\theta''_k]^*_{lm} &= i(\lambda_k)^2\lambda'^*_{km}\lambda'^*_{kl} - i\lambda_k\lambda''^*_{klm} \\
[\theta''_k]^*_{lm} &= i(\lambda_k^*)^2\lambda'_{km}\lambda'_{kl} + i\lambda_k^*\lambda''^*_{klm} - 2i(\lambda_k^*)^2\lambda'_{kl}\lambda'_{km} \\
[\theta''_k]^*_{lm} &= -i(\lambda_k^*)^2\lambda'_{km}\lambda'_{kl} + i\lambda_k^*\lambda''^*_{klm} \\
\therefore [\theta''_k]^*_{lm} &= [\theta''_k]_{lm}
\end{aligned}
\tag{92}
$$

This implies $[\theta''_k]_{lm}$ is a real quantity

Part II

CODING

6 Generating SU(N) matrix from SU(2)

```
! This program generates the SU(N=5) matrix
PROGRAM STPDC
IMPLICIT NONE

nteger,parameter:: NN = 5
Integer::  i,j,k,l,p,q,t,u,s
real(KIND=8):: T1,T2,T3,phii,xi,theta,pi
Complex*16:: SU2(2,2),SUN(NN,NN),SUNM(NN,NN),SUNP(NN,NN),WW(NN,NN)
complex*16:: ai,a_a,b_b,II(NN,NN),WC(NN,NN),UC(NN,NN),III(NN,NN)

    pi = dacos(-1.d0)
    ai = dcmplx(0.d0,1.d0)
                              s = 1
                26          t = s+1

                25           call  random_number(harvest=T1)
                              xi = (pi*(2*T1-1))
          call  random_number(harvest=T2)
         theta =( 0.5*pi*(T2))
          call  random_number(harvest=T3)
         phii = (pi*(2*T3-1))
                a_a = dcos(theta)*(cdexp(ai*phii))
         b_b = dsin(theta)*(cdexp(ai*xi))
                            SU2(1,1) = a_a
                    SU2(1,2) = b_b
                            SU2(2,2) = dconjg(SU2(1,1))
                    SU2(2,1) = -dconjg(SU2(1,2))

                       Do p = 1,NN
                       DO q = 1,NN
                          IF (p . EQ. q) THEN
                         SUN(p,q) = DCMPLX(1.d0,0.d0)
                          ELSE  IF (p . NE. q) THEN
                         SUN(p,q) = DCMPLX(0.d0,0.d0)
                          end  if
                       end  do
                       end  do

                   SUN(s,s) =   SU2(1,1)
                   SUN(s,t) =  SU2(1,2)
                   SUN(t,s) =  SU2(2,1)
                   SUN(t,t) = SU2(2,2)
                                                DO p =1,NN
                                                Do q = 1,NN
                                               UC(p,q) = dconjg(SUN(q,p))
                                                End  do
                                                End  do

                                   II =  matmul(UC,SUN)

                                                DO p =1,NN
                                                Do q = 1,NN
                                               !write(501,*)"II", p,q, II(p,q)
```

```
                                       End  do
                                       End  do
                                      Write(501,*) "space"

           IF (s . eq.1 . and. t . eq. 2) then
             SUNM = SUN
           else
             SUNP =  MATMUL(SUNM,SUN)
             SUNM = SUNP
           end  if

              t = t+1

           if (t . lt. NN+1) then
              go to 25
           else  if (t . eq. NN+1) then
               s = s+1
                if (s . lt. NN) THEN
               go to 26
                ELSE  if (s . eq. NN) then
               WW = SUNM

                                       DO p =1,NN
                                       Do q = 1,NN
                                      write(120,*) WW(p,q)
                                       End  do
                                       End  do

                                       DO p =1,NN
                                       Do q = 1,NN
                       WC(p,q) = dconjg(WW(q,p))
                                       End  do
                                       End  do

                       III =  matmul(WC,WW)

                                       DO p =1,NN
                                       Do q = 1,NN
                                      !write(*,*)"III", p,q, III(p,q)
                                       End  do
                                       End  do

               GO TO 27
                end  if
           end  if



27        END PROGRAM STPDC !******************
```

6.1 Newton-Rapson -Method (NRM)

- **For SU(5)**

```
PROGRAM STPDC
IMPLICIT NONE

INTEGER,PARAMETER:: NN = 5,ITR = 40, rpt = 10000
real,parameter::  LLM = 0.00000000001
Integer:: n,m,i,j,k,o,l,p,q,t,ut,INFO,LWORK,IPIV(NN),NTT(120,NN),NT(NN)
integer::           YC,NC
Integer,parameter::  LDA = NN ,LDAA=NN-1, LWMAX = 10000
REAL(KIND=8)::  S,grad_s,pi, RWORK( 3*NN-2 ),trace,MDIF,r1,r2,r3,r4
REAL(KIND=8),DIMENSION(NN):: CC,NNEBS,new_phi,mthet,n_thet,RTPHI,NNPHI,W,NR
REAL(KIND=8),DIMENSION(NN):: xx,yy,thet,phi,SS,NBPHI,NPHI,NEBS,n_phi
REAL(KIND=8),DIMENSION(NN,NN)::
    HM,THM,NA,NNA,DSD,dif_thet,DS,dCC,dft,CKK,dtheta,HMTHM
REAL(KIND=8),DIMENSION(NN,NN,NN):: DDS,DDS1,DDS2,DDS3,DDS4,DDC,ddtheta
REAL(KIND=8)::
    NDSD(NN-1,NN-1),IDSD(NN-1,NN-1),NNS(NN-1),PDF(121),AAA(2,2),III(2,2)
INTEGER:: VOM(NN,NN,NN,NN),tt
complex*16:: ai,bi,lambda(NN)
Complex*16,dimension(NN,NN)::
    DD,WW,UU,DEL,dlamb,JPT,AC,AK,Bk,CK,AA,AKK,BKK,DGZ1,DGZ2
Complex*16,DIMENSION(NN,NN,NN):: ddlamb1,ddlamb2,ddlamb,U_D,COF
Complex*16,DIMENSION(NN,NN,NN,NN)::U_DD
complex*16:: WORK(LWMAX),x(NN),A(LDA,NN),n_A(NN,NN),n_lambda(NN)
CHARACTER*1:: UPLO



             NC = 0
             YC = 0


            ut = 1

      160   t = 1
            pi = dacos(-1.d0)
            ai = dcmplx(0.d0,1.d0)
            bi = dcmplx(0.d0,1.d0)





            call  random_number(harvest = r1)
              phi(1) = 0.5*pi*(2*r1-1)
            Call  random_number(harvest = r2)
              phi(3) = 0.5*pi*(2*r2-1)
            Call   random_number(harvest= r3)
              phi(2) = 0.5*pi*(2*r3-1)
            Call  random_number(harvest = r4)
```

```
                phi(5) = 0.5*pi*(2*r4-1)
                phi(4) = -(phi(1) + phi(3) + phi(2) + phi(5))



   !Write(555,*) ut,t,phi(1),phi(2),phi(3),phi(4),phi(5)


   Do p = 1,NN
                 DO q = 1,NN
                   DD(p,q) = DCMPLX(0.d0,0.d0)
                    IF (p . EQ. q) THEN
                   DD(p,q) = DD(p,q) + cdexp(-ai*phi(p))
                    ELSE  IF (p . NE. q) THEN
                   DD(p,q) = DD(p,q) + DCMPLX(0.d0,0.d0)
                    end  if
                 end  do
                 end  do


      25    OPEN (unit = 120, file = 'fort.120')
  DO l = 1,NN
  DO m = 1,NN
 READ(120,*) WW(l,m)
 !PRINT*,l,m,WW(l,m)
  End  do
  END  DO
 CLOSE(unit = 120)

             UU =  MATMUL(DD,WW)


             Do p = 1,NN
             DO q = 1,NN
                A(p,q) =dcmplx(0.d0,0.d0)
                A(p,q) = A(p,q) + (UU(p,q) + dCONJG(UU(q,p)))*0.5
             END  DO
             END  DO

              AA = A

             !Lapack-for eigen value
             LWORK = -1
              CALL ZHEEV( "V", "L", NN, A, LDA, W, WORK, LWORK, RWORK, INFO )
             LWORK =  min( LWMAX,  INT( WORK( 1 ) ) )
              CALL ZHEEV( "V", "L", NN, A, LDA, W, WORK, LWORK, RWORK, INFO )

           IF( INFO. GT.0 ) THEN
          WRITE(*,*)'The algorithm failed to compute eigenvalues.'
           else  if(INFO. EQ.0) THEN
           Do p = 1,NN
          !WRITE(*,*)t, p,"th eigen value => ",w(p)
          !write(*,*)t,p,"th-eigen-vector=>"
           Do q = 1,NN
          !write(*,*) t,"A(",q,p,")=>",A(q,p)
           End  do
```

```
                 end  do
                 END  IF
                 do p = 1,NN
                 do q = 1,NN
                A(q,p) = A(q,p)
                 end  do
                 end  do

                     Do k = 1,NN
                    lambda(k) = dcmplx(0.d0,0.d0)
                     Do p = 1,NN
                     Do q = 1,NN
                    lambda(k) = lambda(k) + (dconjg(A(p,k))*UU(p,q)*A(q,k))
                     End  do
                     End  do
                    !print*,(dconjg(lambda(k))*lambda(k))
                     End  do


!CALL THETT(NN,XX,YY,THET,LAMBDA,t)
                 Do k = 1,NN
                thet(k) = ai*cdlog(lambda(k))
                 end  do
                              Do k = 1,NN
                             thet(k) = -ai*cdlog(cdexp(ai*thet(k)))
                              end  do

!Call REARR(NN,dft,PDF,NT,MDIF,n_lambda,n_thet,lambda,thet,n_A,A,t,phi)

             Do l = 1,NN
             Do m = 1,NN
            dft(l,m) = 0.d0
            dft(l,m) = dft(l,m) + dabs(phi(l) - thet(m))
             end  do
             end  do

            tt = 0

              DO k = 1,NN
              do l = 1,NN
              do m = 1,NN
              do n = 1,NN
              do o = 1,NN


                     if (k . ne. l) then
                      if (k . ne. m) then
                       if (k . ne. n) then
                        if (k . ne. o) then
                         if (l . ne. m) then
                          if (l . ne. n) then
                           if (l . ne. o) then
                            if (m . ne. n) then
                             if (m . ne. o) then
                              if (n . ne. o) then
```

```
           tt = tt+1
          !print*,tt, k,l,m,n,o

           NTT(tt,1) = k
           NTT(tt,2) = l
           NTT(tt,3) = m
           NTT(tt,4) = n
           NTT(tt,5) = o

           PDF(tt) = dsqrt((dft(1,NTT(tt,1)))**2 + (dft(2,NTT(tt,2)))**2 &
             & + (dft(3,NTT(tt,3)))**2 + (dft(4,NTT(tt,4)))**2 &
               & +  (dft(5,NTT(tt,5)))**2)

          ! print*,"pdf",tt,PDF(tt)

                        end  if
                          end  if
                            end  if
                              end  if
                                end  if
                                  end  if
                                   end  if
                                     end  if
                                       end  if
                                         end  if

                      end  do
                 end  do
          end  do
    end  do
end  do





   Do tt = 1,120
       IF (tt . eq. 1) then
      MDIF = PDF(1)
       else
      MDIF =  MIN(PDF(tt),MDIF)
       end  if
   END  DO



    do tt = 1,120
      if (MDIF . eq. PDF(tt)) then
        do l = 1,NN
         NT(l) = NTT(tt,l)
          end  do
           end  if
            end  do

       !print*,MDIF,NT(1),NT(2),NT(3),NT(4),NT(5)
```

```
             do l = 1,NN
               n_lambda(l) = lambda(NT(l))
             end  do
             do l = 1,NN
               n_thet(l) = thet(NT(l))
             end  do
             DO l = 1,NN
             DO m = 1,NN
               n_A(m,l) = A(m,NT(l))
             end  do
             END  DO

            !newly arranged thet and eigen vectors------------
             do l = 1,NN
            lambda(l) = n_lambda(l)
             end  do
             do l = 1,NN
            thet(l) = n_thet(l)

             end  do
            !write(207,*) thet(1),thet(2),thet(3),thet(4),thet(5)
             do l = 1,NN
             do m = 1,NN
            A(l,m) = n_A(l,m)
             end  do
             end  do

!CALL UDAS(t,NN,DEL,UU,U_D,ai)
                    do p= 1,NN
                    do q = 1,NN
                       if (p . eq.q) then
                      DEL(p,q) = dcmplx(1.d0,0.d0)
                       else  if(p . ne. q) then
                      DEL(p,q) = dcmplx(0.d0,0.d0)
                       end  if
                   end  do
                   end  do

                  ai = dcmplx(0.d0,1.d0)
                   DO l = 1,NN
                   Do p= 1,NN
                   DO q = 1,NN
                   U_D(l,p,q) = dcmplx(0.d0,0.d0)
                   U_D(l,p,q) = U_D(l,p,q) -( (ai)*DEL(p,l)*UU(p,q))
                   !write(109,*) l,p,q,U_D(l,p,q)
                        end  Do
                        end  do
                        End  do

!CALL UDDAS(t,NN,DEL,UU,U_DD)
                   DO l = 1,NN
                   DO m = 1,NN
                   Do p= 1,NN
```

```
                    DO q = 1,NN
                    U_DD(l,m,p,q) = dcmplx(0.d0,0.d0)
                    U_DD(l,m,p,q) = U_DD(l,m,p,q) &
                    &-( DEL(p,l)*DEL(p,m)*UU(p,q))
                         end  Do
                         end  do
                         End  do
                         END  DO
! CALL LLAMB(t,NN,dlamb,A,U_D)
                    Do l = 1, NN
                    Do k = 1, NN
                   dlamb(k,l) = dcmplx(0.d0,0.d0)
                    Do p = 1,NN
                    Do q = 1,NN
                   dlamb(k,l) = dlamb(k,l) + &
                   &(dconjg(A(p,k))*U_D(l,p,q)*A(q,k))
                    End  do
                    end  do
                    end  do
                    end  do
 !CALL COFF(t,NN,A,U_D,lambda,COF)
                   Do l = 1,NN
                   DO k = 1,NN
                   DO q = 1,NN
                   COF(l,k,q) = DCMPLX(0.D0,0.D0)
                      IF (k . ne. q) then
                        DO i = 1,NN
                        DO j = 1,NN
                        COF(l,k,q) = COF(l,k,q) +&
                         & (((dconjg(A(i,q)))*U_D(l,i,j)*A(j,k))&
                         &/(lambda(k)-lambda(q)))
                        END  do
                        end  do
                     end  if
                    END  do
                    END  DO
                    end  do
!CALL GLAMBDA(t,NN,lambda,ddlamb1,ddlamb2,ddlamb,A,U_DD,COF)
                    !dlamb1--------------------------------
                    DO k = 1,NN
                    Do l = 1,NN
                    Do m = 1,NN
                   ddlamb1(k,l,m) = dcmplx(0.d0,0.d0)
                        Do p = 1,NN
                        Do q = 1,NN
                       ddlamb1(k,l,m) = ddlamb1(k,l,m) +
                            (dconjg(A(p,k))*U_DD(l,m,p,q)*A(q,k))
                        END  DO
                        END  DO

                    END  DO
                    END  DO
                    END  DO
              !dlamb2----------------------------
                    DO k = 1,NN
                    Do l = 1,NN
                    Do m = 1,NN
```

```
                  ddlamb2(k,l,m) = dcmplx(0.d0,0.d0)
                      Do p = 1,NN
                       If (p. ne.k) then
                       ddlamb2(k,l,m) = ddlamb2(k,l,m) + ((COF(l,p,k)*COF(m,k,p)&
                                    & + COF(m,p,k)*COF(l,k,p))*(lambda(p)-lambda(k)))
                       end  if
                      END  DO
                   END  DO
                   END  DO
                   END  DO
            !dlamb-----------------------------------
                   DO k = 1,NN
                   Do l = 1,NN
                   Do m = 1,NN
                     ddlamb(k,l,m) = dcmplx(0.d0,0.d0)
                     ddlamb(k,l,m) = ddlamb(k,l,m) + (ddlamb1(k,l,m)+ddlamb2(k,l,m))
                     !write(*,*) k,l,m,"=", ddlamb(k,l,m)
                   END  DO
                   END  DO
                   END  DO


!CALL GTHETA(t,NN,bi,dtheta,ddtheta,dlamb,ddlamb,lambda)
                 !dtheta------------------------------
                  bi = dcmplx(0.d0,1.d0)
                   DO k = 1,NN
                   Do l = 1,NN
                  dtheta(k,l) = 0.d0
                  dtheta(k,l) = dtheta(k,l) + &
                  & bi*dconjg(lambda(k))*dlamb(k,l)
                   end  do
                   end  do
              !ddtheta---------------------------
                   DO k = 1,NN
                   Do l = 1,NN
                   DO m = 1,NN
                  ddtheta(k,l,m) = 0.d0
                  ddtheta(k,l,m) = ddtheta(k,l,m)
                       -(bi*((dconjg(lambda(k)))**2)*(dlamb(k,m)*dlamb(k,l))) &
                                  & + (bi*(dconjg(lambda(k))*ddlamb(k,l,m)))
                   end  do
                   end  do
                   END  DO







 !ACTION-PART!!
 !CALL NEBLA_S(t,NN,cc,ss,ds,thet,phi,dtheta,DEL)
                          !cc----------------------
                                 DO i = 1,NN
                                cc(i) = 0.d0
                                cc(i) = cc(i) + (1/(thet(i)-phi(i)))
```

```
                 END  DO
        !ss--------------------------
                 DO i = 1,NN
                ss(i) = 1.d0
                ss(i) = ss(i)*(thet(i)-phi(i))**2
                 END  DO
        !ds----------------------------
                 Do i = 1,NN
                 Do l = 1,NN
                ds(i,l) = 0.d0
                ds(i,l) = ds(i,l) +&
                & (2.d0*(cc(i)*ss(i)*(dtheta(i,l) - DEL(i,l))))
                 end  do
                 end  do




  !DCC-------------------------
         DO i = 1,NN
         DO m = 1,NN
          dcc(i,m) = 0.d0
          dcc(i,m) = dcc(i,m) - ((cc(i))**2)*(dtheta(i,m)-DEL(i,m))
         END  DO
         END  DO

 !(DDS1)-----------------------
         DO i = 1,NN
         DO l = 1,NN
         DO m = 1,NN
          DDS1(i,l,m) = 0.d0
          DDS1(i,l,m) = DDS1(i,l,m) +&
          & (2.d0*ds(i,m)*cc(i)*(dtheta(i,l)))
         END  DO
         END  DO
         END  DO
 !(DDS2)---------------------------
         DO i = 1,NN
         DO l = 1,NN
         DO m = 1,NN
          DDS2(i,l,m) = 0.d0
          DDS2(i,l,m) = DDS2(i,l,m) + &
          &(2.d0*SS(i)*DCC(i,m)*(dtheta(i,l)))
         END  DO
         END  DO
         END  DO
!(DDS3)----------------------------------------
         DO i = 1,NN
         DO l = 1,NN
         DO m = 1,NN
          DDS3(i,l,m) = 0.d0
          DDS3(i,l,m) = DDS3(i,l,m) + (2.d0*ss(i)*CC(i)*(ddtheta(i,l,m)))
         END  DO
         END  DO
         END  DO
!(DDS4)----------------------------------
         DO i = 1,NN
```

```
                         DO l = 1,NN
                         DO m = 1,NN
                          DDS4(i,l,m) = 0.d0
                          DDS4(i,l,m) = DDS4(i,l,m) +&
                          &((2.d0*DS(i,m)*cc(i)*DEL(i,l)) +&
                          & (2.d0*ss(i)*dcc(i,m)*DEL(i,l)))
                         END  DO
                         END  DO
                         END  DO
          !(DDS)----------------------------------------
                         DO i = 1,NN
                         DO l = 1,NN
                         DO m = 1,NN
                        DDS(i,l,m) = 0.d0
                        DDS(i,l,m) = DDS(i,l,m) + &
                        &(DDS1(i,l,m)+ DDS2(i,l,m)+ &
                        & DDS3(i,l,m) - DDS4(i,l,m))
                         END  DO
                         END  DO
                         END  DO
           !H-matrix----------------------------------------
                        DO p = 1,NN
                        DO q = 1,NN
                       DSD(p,q) = 0.d0
                          Do i = 1,NN
                         DSD(p,q) = DSD(p,q) + DDS(i,p,q)
                          end  do
                       !WRITE(*,*) "DSD(",p,q,")=", DSD(p,q)
                        end  do
                        end  do
!CALL SNGREDS(NN,S,ss,NEBS,ds,grad_s,t)
                    !S-----------------------------------------------
                    S = 0.d0
                     Do i = 1,NN
                    S = S + ss(i)
                     END  DO
                    !Write(111,*) t,S
            !NEB-S----------------------------------
                     Do k = 1,NN
                    NEBS(k) = 0.d0
                     Do l = 1,NN
                    NEBS(k) = NEBS(k) + ds(l,k)
                     END  DO
                     END  DO
                    grad_s = 0.d0
                     DO P = 1,NN
                    grad_s = grad_s + NEBS(p)**2
                     end  do
                    grad_s = dsqrt(grad_s)

                    !write(112,*) t,grad_s

 !CARTANS METHOD OF TRANSFORMATION

      DO P = 1,NN
      DO q = 1,NN
      IF (p . lt. NN)then
```

```
         if (q . eq. p) then
        HM(p,q) = (1.d0/dsqrt(dfloat(p*(p+1))))
         ELSE  If (q . lt. p) then
        HM(p,q) = (1.d0/dsqrt(dfloat(p*(p+1))))
         else  If ( q . EQ. p+1) then
        HM(p,q) = ((0.d0 - dfloat(p))/dsqrt(dfloat(p*(p+1))))
         else  IF (q . gt. p+1) then
        HM(p,q) = (0.d0)
         end  if
         ELSE  IF (P . EQ. NN) THEN
        HM(p,q) = (1.d0/dsqrt(dfloat(p*(p+1))))
      END  IF
     End  do
     END  DO


    !NEW-NEBS-------------------------------
    Do  p = 1,NN
   NNEBS(p) = 0.d0
    Do l= 1,NN
    NNEBS(p) = NNEBS(p) + HM(p,l)*NEBS(l)
    end  do
    end  do

!A-----------!REGAIN NEBS=C-----------------
       DO k= 1,NN-1
       NNS(k) = NNEBS(k)
       END  DO


   !NEW-NA-NNA-------------------------------
     DO q = 1,NN
    THM(p,q) = HM(q,p)
     end  do
     end  do

  !NA= MATMUL(A,HM)---------

     DO p = 1,NN
     DO q = 1,NN
      NA(p,q) = 0.d0
       do l = 1,NN
      NA(p,q) =NA(p,q) + (DSD(p,l)*THM(l,q))
       end  do
    end  do
    end  do

   !A = MATMUL(THM,NA)-------------------------

    DO p = 1,NN
    DO q = 1,NN
      NNA(p,q) = 0.d0
       do l = 1,NN
      NNA(p,q) =NNA(p,q) + (HM(p,l)*NA(l,q))
       end  do
    end  do
    end  do
```

```
    DO p = 1,NN-1
    Do q = 1,NN-1
   NDSD(p,q) = NNA(p,q)
    END  DO
    END  DO
                    !lapack inversion of DSD
                      CALL DGETRF( NN-1,NN-1,NDSD,LDAA,IPIV,INFO)
                      If (info . eq.0) then
                   LWORK = -1
              CALL DGETRI( NN-1, NDSD, LDAA, IPIV, WORK,LWORK, INFO )
               LWORK =  min( LWMAX,  INT( WORK( 1 ) ) )
              CALL DGETRI( NN-1, NDSD, LDAA, IPIV, WORK, LWORK, INFO )
              if (info . ne. 0) then
             PRINT*, 'Matrix inversion failed!'
              end  if
              end  if


!CALL FNLPHI (t,NN,IDSD,NDSD,NBPHI,NNPHI,NNS,RTPHI,THM,new_phi,phi,DD,trace,ITR,III,AAA)
              DO l = 1,NN-1
       Do m = 1,NN-1
      IDSD(l,m) = NDSD(l,m)
       End  do
       End  do
    !NEB-PHI-------------------------------------------------------

       DO p = 1,NN-1
      NBPHI(p) = 0.d0
       DO i= 1,NN-1
      NBPHI(p) = NBPHI(p) + IDSD(p,i)*NNS(i)
       end  do
       end  do
   !Final new-phi--------------------------------------------------
              Do p = 1,NN
                 if (p . lt.NN) then
                NNPHI(p) = NBPHI(p)
                 else  if (p . eq. NN) then
                NNPHI(p) = 0.d0
                 END  IF
              end  do

    !RTPHI = matmul(THM,NNPHI)
              DO l = 1,NN
      RTPHI(l) = 0.d0
       DO i= 1,NN
      RTPHI(l) = RTPHI(l) + THM(l,i)*NNPHI(i)
       end  do
       end  do

              do p = 1,NN
             new_phi(p) = 0.d0
             new_phi(p) = new_phi(p) + (phi(p) - RTPHI(p))
              end  do
```

```
          do p = 1,NN
         new_phi(p) = -ai*cdlog(cdexp(ai*new_phi(p)))
          end  do



!redefine new set of phi----------------------------------------
                      do p = 1,NN
                     phi(p) = 0.d0
                     phi(p) = phi(p) + new_phi(p)
                      end  do

                     print*,ut,phi(1),phi(2),phi(3)!,phi(4) ,phi(5)


      Do p = 1,NN
                      DO q = 1,NN
                        DD(p,q) = DCMPLX(0.d0,0.d0)
                         IF (p . EQ. q) THEN
                        DD(p,q) = DD(p,q) + cdexp(-ai*new_phi(p))
                         ELSE  IF (p . NE. q) THEN
                        DD(p,q) = DD(p,q) + DCMPLX(0.d0,0.d0)
                         end  if
                      end  do
                      end  do

                        trace = phi(1) + phi(2) + phi(3) + phi(4) + phi(5)
                        !write(666,*)ut,t,"trace=",trace


                t  = t+1

                   If (t . gt. ITR) then
               !print*,"NO CONVERGENCE"
               !write(555,*)"space"
               print*, "space"
               NC = NC + 1
               ut = ut + 1
                      if (ut . lt. rpt+1) then
                     go to 160
                      else  if (ut . eq. rpt+1) then
                     go to 28
                      end  if
                end  if


                if (S . gt. LLM ) then
               GO TO 25
                else  if (S . lt. LLM ) then

                     YC = YC +1
                     ut = ut + 1

                     print*,"space"
                     !write(555,*)"space"
```

```
                    write(99,*)phi(1),phi(2),phi(3),phi(4),phi(5)
                     if (ut . lt. rpt+1) then
                    go to 160
                     else  if (ut . eq. rpt+1) then
                    go to 28
                     end  if

               end  if


            28  write(1983,*) "Yes convergence = ",YC
                write(1983,*) "No convergence = ",NC

                WRITE(1983,*) "Efficiency(%) =", ( float(YC)/ float(RPT))*100



              END PROGRAM STPDC !******************
```

6.2 Ploting a Polygon

```
! This program plots the polygon

 PROGRAM STPDC
 IMPLICIT NONE

 Integer,parameter::         NN = 3, NR = 6
 Integer::                 i,j,k,l,p,q,t,s,NT,m
 real(KIND = 8)::              phi(NR,NN),phii(NR,NN),trace(NR),tr
 real(KIND = 8)::             D(NR),DD,lv(NR),e(NN,NN),u(NN,NN),v(NN,NN)





               OPEN (unit = 1000, file = 'fort.1000')
                do l = 1,NR
        READ(1000,*) phi(l,1),phi(l,2),phi(l,3)
                end  do
   CLOSE(unit = 1000)


               Do l = 1,NR
               Do k = 1,NN
              phii(l,k) = 0.d0
              phii(l,k) = phii(l,k) + (phi(l,k) - phi(1,k))
              !print*,l,k,phii(l,k)
               end  do
              trace(l) = phii(l,1) + phii(l,2) + phii(l,3)
              !print*,l,trace(l)
```

```
        end  do



        Do l = 2,NR
        do k = 1,NN
       D(l) = 0.d0
       D(l) = D(l) + (phii(1,k) - phii(l,k))**2
        end  do
       !print*,l,D(l)
        end  do


       DD =  Min(D(2),D(3),D(4),D(5),D(6))

        IF (D(2) . eq. DD) then
           NT = 2
        Else   IF (D(3) . eq. DD) then
           NT = 3
        Else   IF (D(4) . eq. DD) then
           NT = 4
        Else   IF (D(5) . eq. DD) then
           NT = 5
        Else   IF (D(6) . eq. DD) then
           NT = 6
        End  if

       !print*,"Nt", NT

        DO k = 1,NN
       v(1,k) = phii(NT,k)
       !print*,"v","1",k,v(1,k),phii(1,k)
        end  do

       lv(1) = dsqrt(v(1,1)**2 + v(1,2)**2 + v(1,3)**2)

        Do k = 1,NN
       e(1,k) = v(1,k)/(lv(1))
        End  do

     u(1,1) = lv(1)
    u(1,2) = 0.d0
     u(1,3) = 0.d0
     do l = 1,NN
   print*,"1",l,u(1,l)
     end  do

        Do k = 1,NN
       e(3,k) = 1.d0/dsqrt(3.d0)
        End  do

      e(2,1) = e(3,2)*e(1,3) - e(3,3)*e(1,2)
      e(2,2) = e(3,3)*e(1,1) - e(3,1)*e(1,3)
      e(2,3) = e(3,1)*e(1,2) - e(3,2)*e(1,1)


       do m = 1,NR
```

```
 do l = 1,NN
u(m,l) = 0.d0
 do k = 1,NN
u(m,l) = u(m,l) + (phii(m,k)*e(l,k))
 end  do
print*,m,l,u(m,l)
 end  do
write(100,*) u(m,1),u(m,2)!,u(m,3)
 end  do
 END PROGRAM STPDC
```

Part III

Side Product

7 Find minimum point of a paraboloid: NRM

```
!This program finds the minimum point of paraboloid by using NRM
PROGRAM STPDC
IMPLICIT NONE

INTEGER,PARAMETER::         NN = 3,ITR = 20
INTEGER::                   I,J,K,L,M,P,Q,t
REAL::                      IDSD(NN,NN),NEBF(NN),X0(NN),X(NN),FF,XX(NN)


                  !f(x,y,z) = (x-1)^3 + (y-2)^3 + (z-3)^3
                    t = 0
                    x0(1) = 100.0
                    x0(2) = 100.0
                    x0(3) = 100.0

                25   DO k = 1,NN
                       NEBF(k) = (3*(x0(k)-k)**2)
                     end do


                     DO l = 1,NN
                     DO M = 1,NN
                    IDSD(l,m) = 0.0
                     if (l . eq. m) then
                         IDSD(l,m) = IDSD(l,m) + (1/(6*(x0(l)-l)))
                     else if (l . ne. m) then
                         IDSD(l,m) = 0.0
                     end if
                    !print*,"IDSD",l,m,IDSD(L,M)
                     END DO
                     END DO

                     DO p = 1,NN
                       x(p) = 0.0
                       xx(p) = 0.0
                     DO l = 1,NN
                     xx(p) = xx(p) - (IDSD(p,l)*NEBF(l))
                     END DO
                     x(p) = x(p) + (x0(p) + xx(p))
                    print*,"x(",p,")=" ,x(p)
                     END DO

                    FF = (x(1)-1)**2+(x(2)-2)**2 + (x(3)-3)**2
                    print*, t,"FF=",FF

                     DO K = 1,NN
                     x0(k) = x(k)
                     END DO

                    t = t+1
                     if (t . lt. itr) then
                    go to 25
                     else if (t. eq. itr) then
                    go to 26
```

```
              end  if


         END PROGRAM STPDC !***************
```

8 Newton Raphson and Cartan's: NRCM

```
! This program find the minimum of paraboloid using Newton rapson and cartan combined.
!There is an interesting concept of rotation
  PROGRAM STPDC
  IMPLICIT NONE

  INTEGER,PARAMETER::          NN = 3,ITR = 30
  INTEGER::                    I,J,K,L,M,P,Q,t
  REAL::                       IDSD(NN,NN),NEBF(NN),XO(NN),X(NN),FF,XX(NN),F(NN)
  REAL::                       DF(NN,NN),DDF(NN,NN,NN),DFT(NN),DDFT(NN,NN)
  REAL,DIMENSION(NN,NN)::      HM,THM,TDDF,NA

  Integer::                INFO,LWORK,IPIV(NN),NT(NN)
  Integer,parameter::        LDAA=NN-1, LWMAX = 10000
  REAL::                       pi, RWORK( 3*NN-2 ),AAA(NN-1,NN-1)

  REAL,DIMENSION(NN)::         NX,NNX,RTX,new_X,TDF

  REAL::                       PDDF(NN-1,NN-1),IDDF(NN-1,NN-1),PDF(NN-1),III(2,2)


  REAL::                        WORK(LWMAX),A(LDAA,NN)
  CHARACTER*1::                 UPLO


                    !f(x,y,z) = (x-x0)^2 * (y-x0)^2 * (z-x0)^2
                            !+ (x-y0)^2 * (y-y0)^2 * (z-z0)^2
                            !+ (x-z0)^2 * (y-z0)^2 * (z-z0)^2
                            !constraint:x+y+z=0


                      t = 0
                      x0(1) = 1.2
                      x0(2) = 2.5
                      x0(3) = -3.7
                      x(1) = 3.0
                      x(2) = 4.0
                      x(3) = -7.0
                       do i = 1,NN
                      print*,t,x(i)
                       end  do
                       DO i = 1,NN
                         F(i) = 1.0
```

```
                 DO j = 1,NN
                   F(i) = F(i)*(x(j) - x0(i))**2
                 end  do
                   F(i) = 0.5*f(i)
                 end  do
                Do i = 1,NN
                   FF = 0.0
                   FF= FF + F(i)
                END  DO
               WRITE(91,*)t,"FF=",FF

                Do i = 1,NN
                DO P = 1,NN
                   DF(i,p) = 0.0
                   DF(i,p) = DF(i,p) + (F(i)/(x(p) - x0(i)))
                END  DO
                END  DO
                Do p = 1,NN
                   DFT(p) = 0.0
                    DO i = 1,NN
                   DFT(p) = DFT(p) + DF(i,p)
                    END  DO
                   WRITE(92,*) t,"DFT",p,DFT(p)
                END  DO


                Do i = 1,NN
                DO p = 1,NN
                DO q = 1,NN
                   DDF(i,p,q) =0.0
                   DDF(i,p,q) = DDF(i,p,q) +
                        2.0*(F(i)/((x(p)-x0(i))*(x(q)-x0(i))))
                END  DO
                END  DO
                END  DO

                DO p = 1,NN
                DO q = 1,NN
                   DDFT(p,q) =0.0
                    DO i = 1,NN
                   DDFT(p,q) = DDFT(p,q) + DDF(i,p,q)
                    END  DO
                   !print*,p,q,DDFT(p,q)
                END  DO
                END  DO
 !CARTAN'S METHOD-------------------------------------
  DO P = 1,NN
  DO q = 1,NN
IF (p . lt. NN)then
  if (q . eq. p) then
 HM(p,q) = (1.d0/dsqrt(dfloat(p*(p+1))))
  ELSE  If (q . lt. p) then
 HM(p,q) = (1.d0/dsqrt(dfloat(p*(p+1))))
  else  If ( q . EQ. p+1) then
 HM(p,q) = ((0.d0 - dfloat(p))/dsqrt(dfloat(p*(p+1))))
  else  IF (q . gt. p+1) then
 HM(p,q) = (0.d0)
```

```
          end  if
          ELSE  IF (P . EQ. NN) THEN
         HM(p,q) = (1.d0/dsqrt(dfloat(p*(p+1))))
          END  IF
        End  do
        END  DO



           Do  p = 1,NN
        TDF(p) = 0.d0
       Do l= 1,NN
     TDF(p) = TDF(p) + HM(p,l)*DFT(l)
       end  do
     end  do


           DO k= 1,NN-1
          PDF(k)  = TDF(k)
           END  DO



    DO p = 1,NN
    DO q = 1,NN
   THM(p,q) = HM(q,p)
  end  do
   nd  do

  DO p = 1,NN
    DO q = 1,NN
    NA(p,q) = 0.d0
    do l = 1,NN
 NA(p,q) =NA(p,q) + (DDFT(p,l)*THM(l,q))
    end  do
     end  do
     end  do

     DO p = 1,NN
   DO q = 1,NN
TDDF(p,q) = 0.d0
    do l = 1,NN
TDDF(p,q) =TDDF(p,q) + (HM(p,l)*NA(l,q))
    end  do
     end  do
     end  do

         DO p = 1,NN-1
         Do q = 1,NN-1
        PDDF(p,q) = TDDF(p,q)
            AAA(p,q) = TDDF(p,q)
                !print*,"AAA",AAA(p,q)
             END  DO
                 END  DO


                         CALL SGETRF( NN-1,NN-1,PDDF,LDAA,IPIV,INFO)
```

```
                  If (info . eq.0) then
                 LWORK = -1
             CALL SGETRI( NN-1, PDDF, LDAA, IPIV, WORK,LWORK, INFO )
              LWORK =  min( LWMAX,  INT( WORK( 1 ) ) )
             CALL SGETRI( NN-1, PDDF, LDAA, IPIV, WORK, LWORK, INFO )
             if (info . ne. 0) then
            PRINT*, 'Matrix inversion failed!'
             end  if
             end  if


      DO l = 1,NN-1
      Do m = 1,NN-1
     IDDF(l,m) = PDDF(l,m)
            !print*,"IDDF",IDDF(l,m)
      End  do
      End  do
            III =  matmul(IDDF,AAA)

             DO l = 1,NN-1
      Do m = 1,NN-1
          !WRITE(*,*)"Identity",III(l,m)
      End  do
      End  do






      DO p = 1,NN-1
     NX(p)  = 0.0
      DO i= 1,NN-1
     NX(p)  = NX(p) + IDDF(p,i)*PDF(i)
      end  do
      end  do

                 Do p = 1,NN
                 if (p . lt.NN) then
                NNX(p) = NX(p)
                 else  if (p . eq. NN) then
                NNX(p) = 0.0
                 END  IF
                 end  do



           DO l = 1,NN
      RTX(l)  = 0.0
    DO i= 1,NN
  RTX(l)  = RTX(l) + THM(l,i)*NNX(i)
    end  do
       !print*,t,l,rtx(l)
    end  do

               do p = 1,NN
              new_X(P) = 0.0
```

```
                new_X(p) = new_X(p) + (X(p) - RTX(p))
                 end  do


                        do p = 1,NN
                       X(p) = new_X(p)
                        end  do
                       t = t+1
                        do p = 1,NN
                       print*,t,"x", x(1),x(2),x(3)
                        end  do


                          if (t . lt. itr) then
                         go to 25
                          else  if (t. eq. itr) then
                         go to 26
                          end  if



                   26  END PROGRAM STPDC
                       !************************************************
```

9 NRC for trigonometric function

```
         !f(x,y,z) = sin(x-x0)^2 * sin(y-x0)^2 * sin(z-x0)^2
                               !+ sin(x-y0)^2 * sin(y-y0)^2 * sin(z-y0)^2
                               !+ sin(x-z0)^2 * sin(y-z0)^2 * sin(z-z0)^2
                               !constraint:x+y+z=0
      PROGRAM STPDC
      IMPLICIT NONE

      INTEGER,PARAMETER::          NN = 3,ITR = 30
      INTEGER::                    I,J,K,L,M,P,Q,t
      REAL::                       IDSD(NN,NN),NEBF(NN),X0(NN),X(NN),FF,XX(NN),F(NN)
      REAL::                       DF(NN,NN),DDF(NN,NN,NN),DFT(NN),DDFT(NN,NN)
      REAL,DIMENSION(NN,NN)::      HM,THM,TDDF,NA

      Integer::               INFO,LWORK,IPIV(NN),NT(NN)
      Integer,parameter::       LDAA=NN-1, LWMAX = 10000
      REAL::                       pi, RWORK( 3*NN-2 ),AAA(NN-1,NN-1)

      REAL,DIMENSION(NN)::         NX,NNX,RTX,new_X,TDF,C(NN,NN)

      REAL::                       PDDF(NN-1,NN-1),IDDF(NN-1,NN-1),PDF(NN-1),III(2,2)

      complex::                     ai
      REAL::                        WORK(LWMAX),A(LDAA,NN)
      CHARACTER*1::                 UPLO
```

```
     t = 0
     x0(1) = 0.15
     x0(2) = 0.30
     x0(3) = -0.45
     x(1) = 0.18
     x(2) = 0.36
     x(3) = -0.54
      do i = 1,NN
     print*,t,"x0 =",x0(i)
      end  do
      do i = 1,NN
     print*,t,"x =",x(i)
      end  do
25    DO i = 1,NN
        F(i) = 1.0
      DO j = 1,NN
        F(i) = F(i)*( sin(x(j) - x0(i)))**2
      end  do
      end  do

     Do i = 1,NN
        FF = 0.0
        FF= FF + F(i)
     END  DO
    WRITE(91,*)t,"FF=",FF
     Do i = 1,NN
     DO P = 1,NN
        C(i,p) =(1/( tan(x(p) - x0(i))))
     END  DO
     END  DO

     Do i = 1,NN
     DO P = 1,NN
        DF(i,p) = 0.0
        DF(i,p) = DF(i,p) + (2.0*F(i)*C(i,p))
     END  DO
     END  DO
     Do p = 1,NN
        DFT(p) = 0.0
         DO i = 1,NN
        DFT(p) = DFT(p) + DF(i,p)
         END  DO
        WRITE(92,*) t,"DFT",p,DFT(p)
     END  DO


     Do i = 1,NN
     DO p = 1,NN
     DO q = 1,NN
        DDF(i,p,q) =0.0
         if (p . eq.q) then
```

```
                         DDF(i,p,q) = DDF(i,p,q) + ((2.0*DF(i,q)*C(i,p)) &
                                           &- (2.0*F(i)*(1+C(i,p)**2)))
                          else  if (p. ne.q) then
                         DDF(i,p,q) = DDF(i,p,q) + (2.0*DF(i,q)*C(i,p))
                          end  if
                       END  DO
                       END  DO
                       END  DO

                       DO p = 1,NN
                       DO q = 1,NN
                         DDFT(p,q) =0.0
                          DO i = 1,NN
                         DDFT(p,q) = DDFT(p,q) + DDF(i,p,q)
                          END  DO
                         !print*,p,q,DDFT(p,q)
                       END  DO
                       END  DO
 !CARTAN'S METHOD------------------------------------
  DO P = 1,NN
  DO q = 1,NN
IF (p . lt. NN)then
  if (q . eq. p) then
 HM(p,q) = (1.d0/dsqrt(dfloat(p*(p+1))))
  ELSE  If (q . lt. p) then
 HM(p,q) = (1.d0/dsqrt(dfloat(p*(p+1))))
  else  If ( q . EQ. p+1) then
 HM(p,q) = ((0.d0 - dfloat(p))/dsqrt(dfloat(p*(p+1))))
  else  IF (q . gt. p+1) then
 HM(p,q) = (0.d0)
  end  if
  ELSE  IF (P . EQ. NN) THEN
 HM(p,q) = (1.d0/dsqrt(dfloat(p*(p+1))))
  END  IF
End  do
END  DO



           Do  p = 1,NN
        TDF(p) = 0.d0
           Do l= 1,NN
          TDF(p) = TDF(p) + HM(p,l)*DFT(l)
           end  do
           end  do


   DO k= 1,NN-1
  PDF(k) = TDF(k)
   END  DO



           DO p = 1,NN
           DO q = 1,NN
               THM(p,q) = HM(q,p)
           end  do
```

```
          end  do

          DO p = 1,NN
          DO q = 1,NN
           NA(p,q) = 0.d0
                 do l = 1,NN
           NA(p,q) =NA(p,q) + (DDFT(p,l)*THM(l,q))
                 end  do
              end  do
            end  do

          DO p = 1,NN
          DO q = 1,NN
           TDDF(p,q) = 0.d0
            do l = 1,NN
           TDDF(p,q) =TDDF(p,q) + (HM(p,l)*NA(l,q))
            end  do
          end  do
          end  do

 DO p = 1,NN-1
 Do q = 1,NN-1
PDDF(p,q) = TDDF(p,q)
    AAA(p,q) = TDDF(p,q)
        !print*,"AAA",AAA(p,q)
     END  DO
         END  DO


              CALL SGETRF( NN-1,NN-1,PDDF,LDAA,IPIV,INFO)
              If (info . eq.0) then
           LWORK = -1
        CALL SGETRI( NN-1, PDDF, LDAA, IPIV, WORK,LWORK, INFO )
         LWORK =  min( LWMAX,  INT( WORK( 1 ) ) )
        CALL SGETRI( NN-1, PDDF, LDAA, IPIV, WORK, LWORK, INFO )
        if (info . ne. 0) then
       PRINT*, 'Matrix inversion failed!'
        end  if
        end  if


 DO l = 1,NN-1
 Do m = 1,NN-1
IDDF(l,m) = PDDF(l,m)
        !print*,"IDDF",IDDF(l,m)
 End  do
 End  do
        III =  matmul(IDDF,AAA)

        DO l = 1,NN-1
 Do m = 1,NN-1
      !WRITE(*,*)"Identity",III(l,m)
 End  do
 End  do
```

```
 DO p = 1,NN-1
NX(p) = 0.0
 DO i= 1,NN-1
NX(p) = NX(p) + IDDF(p,i)*PDF(i)
 end do
 end do

            Do p = 1,NN
            if (p . lt.NN) then
           NNX(p) = NX(p)
            else if (p . eq. NN) then
           NNX(p) = 0.0
            END IF
            end do



                   DO l = 1,NN
           RTX(l) = 0.0
            DO i= 1,NN
           RTX(l) = RTX(l) + THM(l,i)*NNX(i)
            end do
                                                     !print*,t,l,rtx(l)
            end do

          do p = 1,NN
         new_X(P) = 0.0
         new_X(p) = new_X(p) + (X(p) - RTX(p))
          end do

                 ai = (0.0,1.0)
                  do p = 1,NN
                 X(p) = -ai*clog(cexp( ai*new_X(p)))
                  end do
                 t = t+1
                 print*,t,"x", x(1),x(2),x(3)


                    if (t . lt. itr) then
                   go to 25
                    else if (t. eq. itr) then
                   go to 26
                    end if


            26 END PROGRAM STPDC
                 !***********************************************
```

10 Checking Taylor Series :Λ

```
   !this program checks the taylor series of lambda(k) in summer research
        2013(flux-problem)

  !Taylor - Lambda

   PROGRAM STPDC
   IMPLICIT NONE

   INTEGER,PARAMETER::        NN = 3,ITR = 1
   Integer::                  m,mm,i,j,k,l,p,q,t,u,INFO,LWORK,IPIV(NN),NT(NN)
   Integer,parameter::         LDA = NN ,LDAA=NN-1, LWMAX = 10000
   REAL(KIND=8)::              S,grad_s,pi, RWORK( 3*NN-2 ),trace,MDIF
   REAL(KIND=8),DIMENSION(NN):: C,NNEBS,new_phi,mthet,n_thet,RTPHI,NNPHI,W,phi0,phi1
   REAL(KIND=8),DIMENSION(NN):: xx,yy,thet,phi,SS,NBPHI,NPHI,NEBS,n_phi
   REAL(KIND=8),DIMENSION(NN,NN)::
        HM,THM,NA,NNA,DC,DSD,dif_thet,dtheta,DS,CC,dft,CKK,WWT,IW
   REAL(KIND=8),DIMENSION(NN,NN,NN):: DDS,DDS1,DDS2,DDS3,DDS4,DDC,ddtheta,DCC
   REAL(KIND=8)::
        NDSD(NN-1,NN-1),IDSD(NN-1,NN-1),NNS(NN-1),PDF(6),AAA(2,2),III(2,2)
   complex*16::
        ai,bi,lambda(NN),NR(NN),lam1(NN),lam0(NN),lam11(NN),lam111(NN)
   Complex*16,dimension(NN,NN)::
        DD,WW,UU,DEL,dlamb,JPT,AC,AK,Bk,CK,AA,AKK,BKK,DGZ1,DGZ2
   Complex*16,DIMENSION(NN,NN,NN):: ddlamb1,ddlamb2,ddlamb,U_D,COF
   Complex*16,DIMENSION(NN,NN,NN,NN)::U_DD
   complex*16::
        WORK(LWMAX),x(NN),A(LDA,NN),n_A(NN,NN),n_lambda(NN),lambdda(NN)
   CHARACTER*1::              UPLO




                  t = 1
                  pi = dacos(-1.d0)
                  ai = dcmplx(0.d0,1.d0)
                  bi = dcmplx(0.d0,1.d0)

     OPEN (unit = 110, file = 'fort.110')
      DO l = 1,NN
     READ(110,*)phi(l)
     !PRINT*,"phi(",l,")=",phi(l)
      End  do
     CLOSE(unit = 110)


     Write(555,*) t,phi(1),phi(2),phi(3)


25       Do p = 1,NN
                       DO q = 1,NN
                         DD(p,q) = DCMPLX(0.d0,0.d0)
```

```
                 IF (p . EQ. q) THEN
                DD(p,q) = DD(p,q) + cdexp(-ai*phi(p))
                 ELSE  IF (p . NE. q) THEN
                DD(p,q) = DD(p,q) + DCMPLX(0.d0,0.d0)
                 end  if
               end  do
               end  do


          OPEN (unit = 120, file = 'fort.120')
 DO l = 1,NN
 DO m = 1,NN
READ(120,*) WW(l,m)
!PRINT*,l,m,WW(l,m)
 End  do
 END  DO
CLOSE(unit = 120)

          !UU = MATMUL(DD,WW)
            Do p = 1,NN
            DO q = 1,NN
             UU(p,q) = dcmplx(0.d0,0.d0)
            Do l = 1,NN
             UU(p,q) = UU(p,q) +(DD(p,l)*(WW(l,q)))
            END  DO
             !print*,"UU",P,Q,UU(p,q)
            END  DO
            END  DO

          Do p = 1,NN
          DO q = 1,NN
             A(p,q) =dcmplx(0.d0,0.d0)
             A(p,q) = A(p,q) + (UU(p,q) + dCONJG(UU(q,p)))*0.5
          END  DO
          END  DO

           AA = A

          !Lapack-for eigen value
          LWORK = -1
           CALL ZHEEV( "V", "L", NN, A, LDA, W, WORK, LWORK, RWORK, INFO )
          LWORK =  min( LWMAX,  INT( WORK( 1 ) ) )
           CALL ZHEEV( "V", "L", NN, A, LDA, W, WORK, LWORK, RWORK, INFO )

          IF( INFO. GT.0 ) THEN
         WRITE(*,*)'The algorithm failed to compute eigenvalues.'
          else  if(INFO. EQ.0) THEN
          Do p = 1,NN
         !WRITE(*,*)t, p,"th eigen value => ",w(p)
         !write(*,*)t,p,"th-eigen-vector=>"
          Do q = 1,NN
         !write(*,*) t,"A(",q,p,")=>",A(q,p)
          End  do
          end  do
          END  IF
          Do p=1,NN
          Do q =1,NN
```

```
          A(q,p) = A(Q,P)
         end  do
         end  do

        !check--------------------------------------------------
         do p = 1,NN
         do q = 1,NN
        AC(p,q) = dconjg(A(q,p))
         end  do
         end  do
         do p = 1,NN
         do q = 1,NN
        JPT(p,q) = 0.D0
         DO l= 1,NN
        JPT(p,q) = JPT(p,q) + (AC(P,l)*A(l,q))
          END  DO
         !PRINT*,"jpt",t,JPT(P,Q)
          END  DO
          END  DO



            Do k = 1,NN
           lambda(k) = dcmplx(0.d0,0.d0)
            Do p = 1,NN
            Do q = 1,NN
           lambda(k) = lambda(k) + (dconjg(A(p,k))*UU(p,q)*A(q,k))
            End  do
            End  do
          !print*,"lambda",lambda(k)
           !print*,t,(dconjg(lambda(k))*lambda(k))
            End  do


                   If (t . eq. itr) then
                   do p = 1,NN
                   do q = 1,NN
                  WRITE(320,*) A(p,q)
                   END  DO
                   END  DO
                   END  IF

!thett--------------------------------------------------------------------

         Do k = 1,NN
        thet(k) = ai*cdlog(lambda(k))
         end  do
                     Do k = 1,NN
                    !WRITE(207,*) t,"THET(",K,")=",THET(K)
                    thet(k) = -ai*cdlog(cdexp(ai*thet(k)))
                    !WRITE(207,*)"II", t,"THET(",K,")=",THET(K)
                      end  do
                    !print*,"sumthett",thet(1)+thet(2)+thet(3)

                    !Check-------------------------------------------------------------
                       DO k = 1,NN
                     lambdda(k) = cdexp(-ai*thet(k))
```

```
                                !print*,"check",lambdda(k)
                                  End  do


!CALL UDAS(t,NN,DEL,UU,U_D,ai)
                      do p= 1,NN
                      do q = 1,NN
                         if (p . eq.q) then
                        DEL(p,q) = dcmplx(1.d0,0.d0)
                         else  if(p . ne. q) then
                        DEL(p,q) = dcmplx(0.d0,0.d0)
                         end  if
                     end  do
                     end  do

                    ai = dcmplx(0.d0,1.d0)
                     DO l = 1,NN
                     Do p= 1,NN
                     DO q = 1,NN
                     U_D(l,p,q) = dcmplx(0.d0,0.d0)
                     U_D(l,p,q) = U_D(l,p,q) - (ai)*DEL(p,l)*UU(p,q)
                     !write(*,*) l,p,q,U_D(l,p,q)
                           end  Do
                           end  do
                           End  do

!CALL UDDAS(t,NN,DEL,UU,U_DD)
                     DO l = 1,NN
                     DO m = 1,NN
                     Do p= 1,NN
                     DO q = 1,NN
                     U_DD(l,m,p,q) = dcmplx(0.d0,0.d0)
                     U_DD(l,m,p,q) = U_DD(l,m,p,q) - DEL(p,l)*DEL(p,m)*UU(p,q)
                     !write(*,*) l,m,p,q,U_DD(l,m,p,q)
                           end  Do
                           end  do
                           End  do
                           END  DO
! CALL LLAMB(t,NN,dlamb,A,U_D)
                     Do k = 1, NN
                     Do l = 1, NN
                    dlamb(k,l) = dcmplx(0.d0,0.d0)
                     Do p = 1,NN
                     Do q = 1,NN
                    dlamb(k,l) = dlamb(k,l) + dconjg(A(p,k))*U_D(l,p,q)*A(q,k)
                     End  do
                     end  do
                    !print*,k,l,"dlamb",dlamb(k,l)
                     end  do
                     end  do


 !CALL COFF(t,NN,A,U_D,lambda,COF)
                    Do l = 1,NN
                    DO k = 1,NN
                    DO q = 1,NN
                    COF(l,k,q) = DCMPLX(0.D0,0.D0)
```

```fortran
                  IF (k . ne. q) then
                   DO i = 1,NN
                   DO j = 1,NN
                   COF(l,k,q) = COF(l,k,q) +&
                    & (((dconjg(A(i,q)))*U_D(l,i,j)*A(j,k))/(lambda(k)-lambda(q)))
                   END  do
                   end  do
                 end  if
                END  do
                END  DO
                end  do

               !write(*,*) COF(1,1,3),-dconjg(COF(1,3,1))

!CALL GLAMBDA(t,NN,lambda,ddlamb1,ddlamb2,ddlamb,A,U_DD,COF)
                !dlamb1---------------------------------------------------------
                DO k = 1,NN
                Do l = 1,NN
                Do m = 1,NN
               ddlamb1(k,l,m) = dcmplx(0.d0,0.d0)
                   Do p = 1,NN
                   Do q = 1,NN
                  ddlamb1(k,l,m) = ddlamb1(k,l,m) +
                       (dconjg(A(p,k))*U_DD(l,m,p,q)*A(q,k))
                   END  DO
                   END  DO

                END  DO
                END  DO
                END  DO
          !dlamb2-----------------------------------------------------------------
                DO k = 1,NN
                Do l = 1,NN
                Do m = 1,NN
               ddlamb2(k,l,m) = dcmplx(0.d0,0.d0)
                  Do p = 1,NN
                   If (p. ne.k) then
                   ddlamb2(k,l,m) = ddlamb2(k,l,m) + ((COF(l,p,k)*COF(m,k,p)&
                                  & + COF(m,p,k)*COF(l,k,p))*(lambda(k)-lambda(p)))
                   end  if
                  END  DO
                END  DO
                END  DO
                END  DO
        !dlamb------------------------------------------------------------------------
                DO k = 1,NN
                Do l = 1,NN
                Do m = 1,NN
                  ddlamb(k,l,m) = dcmplx(0.d0,0.d0)
                  ddlamb(k,l,m) = ddlamb(k,l,m) + (ddlamb1(k,l,m)+ddlamb2(k,l,m))
                  !write(*,*) k,l,m,"=", ddlamb(k,l,m)
                END  DO
                END  DO
                END  DO

        !Hello! check taylor
             series-------------------------------------------------------------------------------
```

```
 if (t . EQ. 1) then
   DO k = 1,NN
    lam0(k) = lambda(k)
    phi0(k) = phi(k)
    !print*,t,k,phi0(k),lambda(k),lam0(k)
   end  do
 end  if


 if (t . EQ. 2) then
   DO k = 1,NN
    lam1(k) = lambda(k)
    phi1(k) = phi(k)
    !print*,t,k,phi1(k),lambda(k),lam1(k)
   end  do

    do k = 1,NN
   lam11(k) = 0.d0
    do l = 1,NN
   lam11(k) = lam11(k) +(dlamb(k,l)*(phi1(l)-phi0(l)))
    end  do
    end  do
    do k = 1,NN
   lam11(k) = lam11(k) + lam0(k)
    end  do


 do k = 1,NN
!print*,k,"lam1=",lam1(k),"lam11=",lam11(k)
 end  do


    do k = 1,NN
   lam111(k) = 0.d0
    do l = 1,NN
    do m = 1,NN
   lam111(k) = lam111(k) + (
        (0.5)*ddlamb(k,l,m)*(phi1(l)-phi0(l))*(phi1(m)-phi0(m)))
    end  do
    end  do
    end  do
    do k = 1,NN
   lam111(k) = lam111(k) + lam11(k)
    end  do




 do k = 1,NN
!print*,k,"lam1=",lam1(k),"lam111=",lam111(k)
write(71,*)lam0(k),lam1(k)
 end  do
!print*,t

 end  if
!end check-------------------------------------------------
```

```
!CALL GTHETA(t,NN,bi,dtheta,ddtheta,dlamb,ddlamb,lambda)
                  !dtheta-----------------------------------------------------
                    bi = dcmplx(0.d0,1.d0)
                     DO k = 1,NN
                     Do l = 1,NN
                    dtheta(k,l) = 0.d0
                    dtheta(k,l) = dtheta(k,l) + bi*dconjg(lambda(k))*dlamb(k,l)
                    write(*,*)k,l, "dtheta=",dtheta(k,l)
                     end  do
                     end  do
               !ddtheta-----------------------------------------------------------
                     DO k = 1,NN
                     Do l = 1,NN
                     DO m = 1,NN
                    ddtheta(k,l,m) = dcmplx(0.d0,0.d0)
                    ddtheta(k,l,m) = ddtheta(k,l,m)
                         -(bi*((dconjg(lambda(k)))**2)*(dlamb(k,m)*dlamb(k,l))) &
                                     & + (bi*(dconjg(lambda(k))*ddlamb(k,l,m)))
                    write(*,*)k,l,m,"ddtheta=",ddtheta(k,l,m)
                     end  do
                     end  do
                     END  DO



                            t = t+1

                     phi(1) = 0.13d0
                     phi(2) = 0.26d0
                     phi(3) = -0.39d0


                      if (t . lt.ITR+1 ) then
                     GO TO 25
                      else  if (t . eq. ITR+1) then
                     go to 28
                      end  if

                     28  END PROGRAM STPDC
                          !*************************************************
```

10.1 Checking Taylor Series: U

```
!Taylor - U
!this program calculates the taylor series of the action of flux problem in summer
!research -2013-july-26
```

```fortran
PROGRAM STPDC
IMPLICIT NONE

INTEGER,PARAMETER::        NN = 3,ITR = 2
Integer::                m,mm,i,j,k,l,p,q,t,u,INFO,LWORK,IPIV(NN),NT(NN)
Integer,parameter::        LDA = NN ,LDAA=NN-1, LWMAX = 10000
REAL(KIND=8)::             S,grad_s,pi, RWORK( 3*NN-2 ),trace,MDIF,phi0(NN)
real(kind=8)::             S0,S1,S11,S111,phi1(NN)
REAL(KIND=8),DIMENSION(NN):: C,NNEBS,new_phi,mthet,n_thet,RTPHI,NNPHI,W,NR
REAL(KIND=8),DIMENSION(NN):: xx,yy,thet,phi,SS,NBPHI,NPHI,NEBS,n_phi
REAL(KIND=8),DIMENSION(NN,NN):: HM,THM,NA,NNA,DC,DSD,dif_thet,dtheta,DS,CC,dft,CKK
REAL(KIND=8),DIMENSION(NN,NN,NN):: DDS,DDS1,DDS2,DDS3,DDS4,DDC,ddtheta,DCC
REAL(KIND=8)::             NDSD(NN-1,NN-1),NNS(NN-1),PDF(6),AAA(2,2),III(2,2)
complex*16::               ai,bi,lambda(NN)
Complex*16,dimension(NN,NN)::
    DD,WW,UU,DEL,dlamb,JPT,AC,AA,DGZ1,DGZ2,U0,U1,U11,U111
Complex*16,DIMENSION(NN,NN,NN):: ddlamb1,ddlamb2,ddlamb,U_D,COF
Complex*16,DIMENSION(NN,NN,NN,NN)::U_DD
complex*16::               WORK(LWMAX),x(NN),A(LDA,NN),n_A(NN,NN),n_lambda(NN)
CHARACTER*1::              UPLO




            t = 1

            phi(1) = 0.012d0
            phi(2) = 0.024d0
            phi(3) = -0.036d0

            pi = dacos(-1.d0)
            ai = dcmplx(0.d0,1.d0)
            bi = dcmplx(0.d0,1.d0)



25  OPEN (unit = 120, file = 'fort.120')
   DO l = 1,NN
   DO m = 1,NN
  READ(120,*) WW(l,m)
  !PRINT*,l,m,WW(l,m)
   End do
   END DO
  CLOSE(unit = 120)

  Write(555,*) t,phi(1),phi(2),phi(3)


        Do p = 1,NN
                    DO q = 1,NN
                      DD(p,q) = DCMPLX(0.d0,0.d0)
                       IF (p . EQ. q) THEN
                      DD(p,q) = DD(p,q) + cdexp(-ai*phi(p))
                       ELSE IF (p . NE. q) THEN
                      DD(p,q) = DD(p,q) + DCMPLX(0.d0,0.d0)
                       end if
```

```
                      end  do
                      end  do

                 UU =  MATMUL(DD,WW)




!CALL UDAS(t,NN,DEL,UU,U_D,ai)
                 do p= 1,NN
                 do q = 1,NN
                    if (p . eq.q) then
                   DEL(p,q) = dcmplx(1.d0,0.d0)
                    else  if(p . ne. q) then
                   DEL(p,q) = dcmplx(0.d0,0.d0)
                    end  if
                end  do
                end  do

               ai = dcmplx(0.d0,1.d0)
                DO l = 1,NN
                Do p= 1,NN
                DO q = 1,NN
                U_D(l,p,q) = dcmplx(0.d0,0.d0)
                U_D(l,p,q) = U_D(l,p,q) - ((ai)*DEL(p,l)*UU(p,q))
                !write(109,*) l,p,q,U_D(l,p,q)
                     end  Do
                     end  do
                     End  do

!CALL UDDAS(t,NN,DEL,UU,U_DD)


                DO l = 1,NN
                DO m = 1,NN
                Do p= 1,NN
                DO q = 1,NN
                U_DD(l,m,p,q) = dcmplx(0.d0,0.d0)
                U_DD(l,m,p,q) = U_DD(l,m,p,q) - (DEL(p,l)*DEL(p,m)*UU(p,q))
                !write(200,*) l,m,p,q,U_DD(l,m,p,q)
                     end  Do
                     end  do
                     End  do
                     END  DO

     !Hello! check taylor
         series---------------------------------------------------------------------------------------------

                if (t . EQ. 1) then
                  U0 = UU
                  phi0 = phi
                end  if


                if (t . EQ. 2) then
                  U1 = UU
                  phi1 = phi
```

```
       do p = 1,NN
       do q = 1,NN
        U11(p,q) = dcmplx(0.d0,0.d0)
         do k = 1,NN
        U11(p,q) = U11(p,q) +(U_D(k,p,q)*(phi1(k)-phi0(k)))
         end  do
       end  do
       end  do

      U11 = U11 + U0


   do p = 1,NN
   do q = 1,NN
  print*,p,q,"U1=",U1(p,q),"U11=",U11(p,q)
   end  do
   end  do

    do p = 1,NN
    do q = 1,NN
       U111(p,q) = dcmplx(0.d0,0.d0)
        do k = 1,NN
        do l = 1,NN
       U111(p,q) = U111(p,q)
            +((0.5d0)*U_DD(k,l,p,q)*(phi1(k)-phi0(k))*(phi1(l)-phi0(l)))
        end  do
        end  do
    end  do
    end  do

     U111 = U111 + U11

   do p = 1,NN
   do q = 1,NN
  print*,p,q,"U1=",U1(p,q),"U111=",U111(p,q)
   end  do
   end  do

   end  if
  !end check-------------------------------------------------

    phi(1) = 0.011d0
    phi(2) = 0.0260d0
    phi(3) = -0.037d0

     t = t+1
    if (t . lt.ITR+1) then
   GO TO 25
    else  if (t . eq. ITR+1) then
   go to 28
    end  if

   28  END PROGRAM STPDC
        !************************************************
```