Routines for the dynamical equations of the period vectors of a crystal under constant external stress

Gang Liu
e-mail: gang.liu@queensu.ca
High Performance Computing Virtual Laboratory
Queen's University, Kingston, Ontario, Canada

(Dated: September 04, 2015)

The purpose of this file is to present source code routines for computing the acceleration of the period vectors of a crystal based on the following dynamical equation

$$\alpha_{\mathbf{h},\mathbf{h}}\ddot{\mathbf{h}} = \left(\overrightarrow{\overrightarrow{\pi}} + \overrightarrow{\overrightarrow{\Upsilon}}\right) \cdot \sigma_{\mathbf{h}} \quad (\mathbf{h} = \mathbf{a}, \mathbf{b}, \mathbf{c}), \tag{1}$$

which is the last one before the "SUMMARY AND DISCUSSION" section in our paper [1, 2].

I. THE FORTRAN 90 VERSION

Here is the module of the interface

```
MODULE PERIOD_ACCELERATION_MDL
   INTERFACE
     SUBROUTINE ACCELERATION_OF_PERIODS( ALPHAS, &
                                CURRENT_PERIODS, &
                                INTERNAL_STRESS, &
                                EXTERNAL_STRESS, &
                           PERIOD_ACCELERATIONS   )
         IMPLICIT NONE
         REAL*8, INTENT(IN) :: ALPHAS(3),            &
                               CURRENT_PERIODS(3,3), &
                               INTERNAL_STRESS(3,3), &
                               EXTERNAL_STRESS(3,3)
         REAL*8, INTENT(OUT):: PERIOD_ACCELERATIONS(3,3)
     END SUBROUTINE ACCELERATION_OF_PERIODS
   END INTERFACE
END MODULE PERIOD_ACCELERATION_MDL
```

where the meaning of each argument is stated by its name.

However better to further emphasis on those in elementary level.

The three elements of the `ALPHA` array are the $\alpha_{\mathbf{h},\mathbf{h}}$s for the period vector $\mathbf{h} = \mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ in sequence.

The sub-arrays `CURRENT_PERIODS(1:3,1)`, `CURRENT_PERIODS(1:3,2)`, and `CURRENT_PERIODS(1:3,3)` are the period vectors $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$ respectively.

The internal stress `INTERNAL_STRESS` array should be calculated as the summation of two terms. One is the kinetic energy term, which is a scalar as in equation (34) of reference [2]. The other is the full interaction dyad, which can be expressed as

$$\overrightarrow{\overrightarrow{\varepsilon}} = -\frac{1}{\Omega} \sum_{\mathbf{z} \in \mathrm{DOF}} \left(\frac{\partial E_{p,MD}}{\partial \mathbf{z}}\right) \mathbf{z}, \tag{2}$$

the last equation of our paper[1, 2]. Be sure here $\mathbf{z}$ is a vector, not a scalar in the z-axis of a usual Cartesian coordinate system. For example, the element `INTERNAL_STRESS(1,2)` should contain the element of the above equation for x-component of $\frac{\partial E_{p,MD}}{\partial \mathbf{z}}$ and y-component of $\mathbf{z}$.

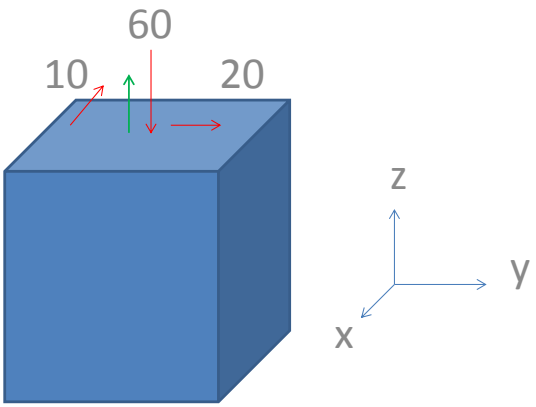

FIG. 1: External stress on top surface of a crystal. The green arrow is the surface direction. The red arrows are the directions of the external forces and the numbers are the corresponding absolute values per unit area.

The external stress `EXTERNAL_STRESS` array is straight-forward. However much attention should be paid to the (positive or negative) sign of each element. For the (red) external forces per unit area acting on the top surface of a crystal as illustrated in Fig. 1, the `EXTERNAL_STRESS` should be set as (with symmetry property considered) $\begin{pmatrix} u & v & -10 \\ v & w & 20 \\ -10 & 20 & -60 \end{pmatrix}$, where u, v, and w should be based on the external forces applied on the side surfaces. When this external stress is applied on the bottom surface of the crystal in Fig. 1, the forces are of the same amplitude but in the opposite directions compared with those on the top surface. When the crystal reaches an equilibrium state, the internal stress must be $\begin{pmatrix} -u & -v & 10 \\ -v & -w & -20 \\ 10 & -20 & 60 \end{pmatrix}$.

The `PERIOD_ACCELERATIONS(1:3,1)`, `PERIOD_ACCELERATIONS(1:3,2)`, and `PERIOD_ACCELERATIONS(1:3,3)` sub-arrays are the accelerations of the period vectors **a**, **b**, and **c** to be returned respectively. That is the purpose of the routine.

The full routine is as follows (next page):

```
SUBROUTINE ACCELERATION_OF_PERIODS( ALPHAS, &
                          CURRENT_PERIODS, &
                          INTERNAL_STRESS, &
                          EXTERNAL_STRESS, &
                     PERIOD_ACCELERATIONS   )
    IMPLICIT NONE
    REAL*8, INTENT(IN) :: ALPHAS(3),            &
                          CURRENT_PERIODS(3,3), &
                          INTERNAL_STRESS(3,3), &
                          EXTERNAL_STRESS(3,3)
    REAL*8, INTENT(OUT):: PERIOD_ACCELERATIONS(3,3)
    REAL*8             :: CELL_SURFACES(3,3)
    REAL*8             :: CELL_VOLUME
    INTEGER            :: I, J, K, L, M, N

    DO I = 1, 3
       J = MOD(I, 3) + 1
       K = MOD(J, 3) + 1
       DO L = 1, 3
          M = MOD(L, 3) + 1
          N = MOD(M, 3) + 1
          CELL_SURFACES(L,I) =                                &
                CURRENT_PERIODS(M,J) * CURRENT_PERIODS(N,K) &
              - CURRENT_PERIODS(N,J) * CURRENT_PERIODS(M,K)
       END DO
    END DO

    CELL_VOLUME = CELL_SURFACES(1,1) * CURRENT_PERIODS(1,1) &
                + CELL_SURFACES(2,1) * CURRENT_PERIODS(2,1) &
                + CELL_SURFACES(3,1) * CURRENT_PERIODS(3,1)

    IF(CELL_VOLUME <= 1.0D-10) THEN
       PRINT*, "Sorry, too small or negative cell volume:"
       PRINT*, CELL_VOLUME
       STOP
    END IF

    DO I = 1, 3
       DO J = 1, 3
          PERIOD_ACCELERATIONS(J, I) = 0.0D0
          DO K = 1, 3
             PERIOD_ACCELERATIONS(J, I) = PERIOD_ACCELERATIONS(J, I) + &
                     (INTERNAL_STRESS(J, K) + EXTERNAL_STRESS(J, K)) * &
                                                 CELL_SURFACES(K, I)
          END DO
          PERIOD_ACCELERATIONS(J, I) = PERIOD_ACCELERATIONS(J, I) &
                                     / ALPHAS(I)
       END DO
    END DO

    RETURN
END SUBROUTINE ACCELERATION_OF_PERIODS
```

II. THE C/C++ VERSION

The interface for C/C++ version is

```
void Acceleration_of_PERIODS( double * alphas,
                              double * current_periods,
                              double * internal_stress,
                              double * external_stress,
                              double * period_accelerations  )
```

where the meaning of each argument is stated by its name. All these pointers point to double (one-dimentional) arrays. The first should have at least three elements and all the rest have at least nine elements.

The meaning in elementary level is as follows.

The three elements of the `alphas` array are the $\alpha_{\mathbf{h},\mathbf{h}}$s for the period vector $\mathbf{h} = \mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ in sequence.

The sub-arrays `current_periods[0:2]`, `current_periods[3:5]`, and `current_periods[6:8]` are the period vectors $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$ respectively, where `an_array[i:j]` means a range of elements from `an_array[i]` to `an_array[j]` throughout this document.

The internal stress `internal_stress` array should be calculated as the summation of two terms. One is the kinetic energy term, which is a scalar as in equation (34) of reference [2]. The other is the full interaction dyad, which can be expressed as

$$\overleftrightarrow{\varepsilon} = -\frac{1}{\Omega} \sum_{\mathbf{z} \in \mathrm{DOF}} \left(\frac{\partial E_{p,MD}}{\partial \mathbf{z}} \right) \mathbf{z}, \tag{3}$$

the last equation of our paper[1, 2]. Be sure here $\mathbf{z}$ is a vector, not a scalar in the z-axis of a usual Cartesian coordinate system. For example, the element `internal_stress[1]` should contain the element of the above equation for x-component of $\frac{\partial E_{p,MD}}{\partial \mathbf{z}}$ and y-component of $\mathbf{z}$.

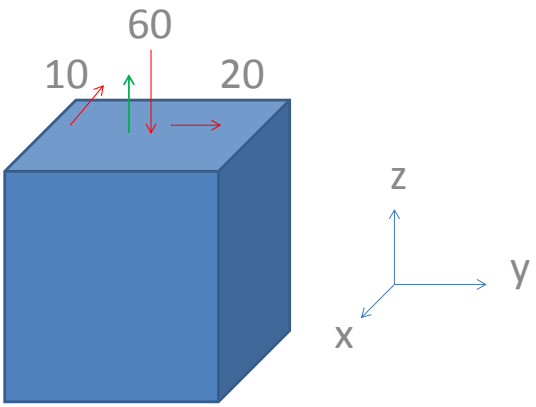

FIG. 2: External stress on top surface of a crystal. The green arrow is the surface direction. The red arrows are the directions of the external forces and the numbers are the corresponding absolute values per unit area.

The external stress `external_stress` array is straight-forward. However much attention should be paid to the (positive or negative) sign of each element. For the (red) external forces per unit area acting on the top surface of a crystal as illustrated in Fig. 2, the `external_stress[0-8]` should be set as (with symmetry property considered) $\begin{pmatrix} u & v & -10 \\ v & w & 20 \\ -10 & 20 & -60 \end{pmatrix}$, where u, v, and w should be based on the external forces applied on the side surfaces. When this external stress is applied on the bottom surface of the crystal in Fig. 2, the forces are of the same amplitude but in the opposite directions compared with those on the top surface. When the crystal reaches an equilibrium state, the internal stress must be $\begin{pmatrix} -u & -v & 10 \\ -v & -w & -20 \\ 10 & -20 & 60 \end{pmatrix}$.

The `period_accelerations[0-2]`, `period_accelerations[3-5]`, and `period_accelerations[6-8]` sub-arrays are the accelerations of the period vectors **a**, **b**, and **c** to be returned respectively. That is the purpose of the routine.

The full routine is as follows (next page):

```
#include <stdio.h>
#include <stdlib.h>


void Acceleration_of_PERIODS( double * alphas,
                              double * current_periods,
                              double * internal_stress,
                              double * external_stress,
                              double * period_accelerations )
{    double cell_surfaces[3][3], cell_volume;
     int               i, j, i3j, k, l, m, n;

     for(i=0; i<3; i++)
        {j = (i+1)%3;
         k = (j+1)%3;
         for(l=0; l<3; l++)
            {m = (l+1)%3;
             n = (m+1)%3;
             cell_surfaces[i][l] =
                 current_periods[j*3+m] * current_periods[k*3+n]
               - current_periods[j*3+n] * current_periods[k*3+m];
            }
        }

     cell_volume = cell_surfaces[0][0] *  current_periods[0]
                 + cell_surfaces[0][1] *  current_periods[1]
                 + cell_surfaces[0][2] *  current_periods[2];

     if(cell_volume < 1.0e-10)
       {printf(" Sorry, too small or negative cell volume: ");
        printf(" %f .\n", cell_volume);
        exit(0);
       }

     for(i=0; i<3; i++)
        {for(j=0; j<3; j++)
            {i3j = i * 3 + j;
             period_accelerations[i3j] = 0.0000000000000000000e0;
             for(k=0; k<3; k++)
                {period_accelerations[i3j]  +=
                    (internal_stress[j*3+k] + external_stress[j*3+k]) *
                                              cell_surfaces[i][k]  ;
                }
             period_accelerations[i3j] = period_accelerations[i3j]
                                       / alphas[i]               ;
            }
        }
}
```

References are in the next page

[1] G. Liu, Can. J. Phys. **93** (9), pages 974-978 (2015), dx.doi.org/10.1139/cjp-2014-0518.
[2] G. Liu, arXiv:cond-mat/0209372v16.