

# Implementation of linear viscoelasticity model in PDLAMMPS

R. Rahman and J.T. Foster  
Center for Simulation, Visualization and  
Real-time prediction (SiViRt)  
The University of Texas at San Antonio  
San Antonio, TX 78249  
Email: rezwanur.rahman@utsa.edu,  
john.foster@utsa.edu

September 12, 2014

## 1 Introduction

In this documentation the discussion is focused on implementation of linear viscoelasticity model in PDLAMMPS [PLPS08],[Pli95]. The peridynamic viscoelasticity formulation (and its time integration) used in this work was developed by John Mitchell at Sandia National Lab [Mit11]. In the PDLAMMPS a new pair-peri-style is added in order to incorporate the viscoelasticity. The new source codes `pair-peri-ves.cpp` and `pair-peri-ves.h` are introduced. Bes `peri-neigh.h` are modified for introducing viscoelasticity.

## 2 Algorithm and implementation

In order to include the viscoelasticity the total extension state can be decomposed into two parts [Mit11] <sup>1</sup>:

---

<sup>1</sup>For detail please see the document on implementation of viscoelasticity in state based peridynamics model [Mit11].

$$\text{Total extension: } e(Y) = e^i(Y) + e^d(Y) \quad (1)$$

$$\text{Volumetric extension: } e^i(Y) = \frac{\theta(Y)|X|}{3} \quad (2)$$

$$\text{Deviatoric extension: } e^d(Y) = |Y| - |X| - \frac{\theta(Y)|X|}{3} \quad (3)$$

Here,  $|Y|$ ,  $|X|$  and  $\theta$  are the reference state, deformation state and dilation state, respectively. The deviatoric extension can be written as:

$$e^d(Y) = e^{de}(Y) + e^{db(i)}(Y) \quad (4)$$

Here,  $e^{de}(Y)$  and  $e^{db(i)}$  are the elastic and back parts of the deviatoric extension. Considering viscoelastic effect the peridynamic force state can be written as:

$$t = t^i + t^d = -\frac{3p}{m}\omega\underline{x} + (\alpha_\infty + \alpha_i)e^d - \alpha_i\omega e^{db} \quad (5)$$

$p$ ,  $k$ ,  $\alpha_i$ ,  $t^i$  and  $t^d$  are hydrostatic pressure, bulk modulus, elastic properties volumetric and deviatoric scalar force states, respectively.  $\alpha_\infty + \alpha_i = \frac{15\mu}{m}$ ; where,  $\mu$ ,  $m$  are the shear modulus and weighted volume, respectively. The influence function:  $\omega(\xi) = \frac{1}{\|\xi\|}$ ,  $\|\xi\|$  is the scalar reference state. At the current or  $t_{n+1}$  timestep the scalar force state based on LPS is:

$$t_{n+1}^i = 3k \left( \frac{\theta(i)}{m(i)}\omega_+ + \frac{\theta(i)}{m(i)}\omega_- \right) e\nu(x_j - x_i) V_j \quad (6)$$

Based on viscoelasticity the deviatoric scalar force state can be written as:

$$\begin{aligned} t_{n+1}^d &= 15(1-\lambda)\mu \left( \frac{\omega_+}{m(i)} + \frac{\omega_-}{m(i)} \right) e\nu(x_j - x_i) V_j \\ &+ 15\mu\lambda \left( \frac{\omega_+}{m(i)} + \frac{\omega_-}{m(i)} \right) (e - e_{n+1}^{db}) \nu(x_j - x_i) V_j \end{aligned} \quad (7)$$

Here,  $V_j$ ,  $e_{n+1}^{db}$ ,  $\nu(x_j - x_i)$  are volume fraction, back extension at current timestep and particle volume scaling factor, respectively.  $\lambda$  varies within zero to one. The back extension at current state can be calculated as <sup>2</sup> [Mit11]:

---

<sup>2</sup><https://software.sandia.gov/trac/peridigm/>

$$e_{n+1}^{db} = (1 - \text{decay}) e_n^d + \text{decay} \cdot e_n^{db} + \beta \Delta e^d \quad (8)$$

$$\Delta e^d = e_{n+1}^d - e_n^d \quad (9)$$

$$\text{decay} = \exp\left(-\frac{dt}{\tau_b}\right) \quad (10)$$

$$\beta = 1 - \frac{\tau_b}{dt} (1 - \text{decay}) \quad (11)$$

At any current timestep (i.e.  $n + 1$ ), for a bond between  $i^{\text{th}}$  and  $j^{\text{th}}$  PD nodes, the deviatoric extension is stored on the fly. The value is stored in `deviatoricextension[i][j]`. Similarly, the back extension is also stored in `deviatoricbackextension[i][j]`. Both of the arrays initialized with zero. In the `fix_peri_neigh.cpp` an integer flag based switch is used in order to trigger the LPS, PMB or VES (Viscoelastic solid) models. The `class` `FixPeriNeigh` looks like:

```
FixPeriNeigh::FixPeriNeigh(LAMMPS *lmp,int narg, char **arg) :
  Fix(lmp, narg, arg)
{

  //Get the pair information
  Pair *anypair01 = force->pair_match("peri/pmb",1);
  Pair *anypair02 = force->pair_match("peri/lps",1);
  Pair *anypair03 = force->pair_match("peri/ves",1);

  isPMB = 0; //Check if PMB
  isLPS = 0; //Check if LPS
  isVES = 0; //Check if VES

  //Select the flag based on which model is being used
  if (anypair01 != NULL)
  {
    isPMB=1;
  }
  else if (anypair02 != NULL)
  {
    isLPS=1;
  }
}
```

```

}
else
{
isVES=1;
}

restart_global = 1;
restart_peratom = 1;
first = 1;

// perform initial allocation of atom-based arrays
// register with atom class
// set maxpartner = 1 as placeholder

maxpartner = 1;
npartner = NULL;
partner = NULL;
if (isVES == 1){
    deviatorextention = NULL;
    deviatorBackextention = NULL;
}
r0 = NULL;
vinter = NULL;
wvolume = NULL;

grow_arrays(atom->nmax);
atom->add_callback(0);
atom->add_callback(1);

// initialize npartner to 0 so atom migration is OK the 1st time

int nlocal = atom->nlocal;
for (int i = 0; i < nlocal; i++) npartner[i] = 0;

// set comm sizes needed by this fix

comm_forward = 1;
}

```

There are three integer flags `isPMB`, `isLPS`, `isVES`. `PMB`, `LPS` and `VES` stand for bond based PD model, linear peridynamic solid and viscoelastic peridynamic solid, respectively. Based on the input file any one of these flags are set to one based on:

```
Pair *anypair01 = force->pair_match("peri/pmb",1);
Pair *anypair02 = force->pair_match("peri/lps",1);
Pair *anypair03 = force->pair_match("peri/ves",1);
```

The stored values are used in:

```
PairPeriVES::compute(int eflag, int vflag)

double PairPeriVES::single(int i, int j, int itype, int jtype,
                           double rsq, double factor_coul,
                           double factor_lj, double &fforce)
```

At any current step the total scalar force state  $t$  is calculated and the current deviatoric extension and back extension are stored in order to use in next step.  $\lambda$  and time constant  $\tau_b$  are material dependent parameters. These are defined in the LAMMPS input script. This will be discussed in the next section. The new LAMMPS pair style `{pair_peri_ves}` is defined with addition arguments in the input file ( $\lambda$  and  $\tau_b$ ):

```
void PairPeriVES::settings(int narg, char **arg)
{
  if (narg) error->all(FLERR,"Illegal pair_style command");
}

/* -----
   set coeffs for one or more type pairs
   ----- */

void PairPeriVES::coeff(int narg, char **arg)
{
  if (narg != 9) error->all(FLERR,"Incorrect args for pair coefficients");
  if (!allocated) allocate();
```

```

int ilo,ihi,jlo,jhi;
force->bounds(arg[0],atom->ntypes,ilo,ihi);
force->bounds(arg[1],atom->ntypes,jlo,jhi);

double bulkmodulus_one = atof(arg[2]);
double shearmodulus_one = atof(arg[3]);
double cut_one = atof(arg[4]);
double s00_one = atof(arg[5]);
double alpha_one = atof(arg[6]);
double mlambdai_one = atof(arg[7]); // New
double mtaui_one = atof(arg[8]); //New

int count = 0;
for (int i = ilo; i <= ihi; i++) {
  for (int j = MAX(jlo,i); j <= jhi; j++) {
    bulkmodulus[i][j] = bulkmodulus_one;
    shearmodulus[i][j] = shearmodulus_one;
    cut[i][j] = cut_one;
    s00[i][j] = s00_one;
    alpha[i][j] = alpha_one;
    m_lambdai[i][j] = mlambdai_one; // New
    m_taubi[i][j] = mtaui_one; // New
    setflag[i][j] = 1;
    count++;
  }
}

if (count == 0) error->all(FLERR,"Incorrect args for pair coefficients");
}

```

### 3 LAMMPS command for PD VES

There is no significant change in the LAMMPS input script for viscoelastic model. For PD VES the LAMMPS commands are:

```

pair_style peri/ves
pair_coeff i j Bulk_modulus Shear_modulus s00 alpha lambda tau

```

The LAMMPS compilation with the VES is same. The updated `Install.sh` should be used.

## 4 Conclusion

The LAMMPS implementation of peridynamic viscoelastic model is still in beta phase. Any bug or issue can be informed to the authors through `rezwanur.rahman@utsa.edu` .

## 5 Acknowledgment

The authors are thankful to Steve Plimpton and S. M. Raiyan Kabir for their useful suggestion regarding the code development for PDLAMMPS implementation of viscoelasticity.

## References

- [Mit11] John A. Mitchell. A nonlocal, ordinary, state-based viscoelasticity model for peridynamics. Technical Report SAND2011-8064, Sandia National Laboratories, May 2011. Available at <https://cfwebprod.sandia.gov/cfdocs/CompResearch/docs/SAND2011-Viscoelast>
- [Pli95] S. Plimpton. Fast parallel algorithms for short-range molecular dynamics. *J Comp Phys*, 117:1–19, 1995. <http://lammps.sandia.gov>.
- [PLPS08] Michael L. Parks, Richard B. Lehoucq, Steven J. Plimpton, and Stewart A. Silling. Implementing peridynamics in molecular dynamics code. *Computer physics communication*, 179:777–783, 2008.