Issue 56

GAṆANAM (गणनम्)

July 2025

HIGH PERFORMANCE COMPUTING NEWSLETTER
INSTITUTE FOR PLASMA RESEARCH, GANDHINAGAR

Inside this issue:
(5 pages)

Simulating the excitation of nonplanar pinned solitons observed in Dusty Plasma Experiments

Prasanta Amat (Research Scholar, IPR)
Email: *prasanta.amat@ipr.res.in*

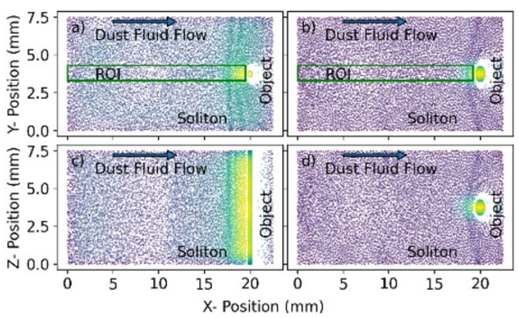


Fig.1 Top view a) and b) and side view c) and d) of the excited cylindrical and spherical pinned solitons, respectively.

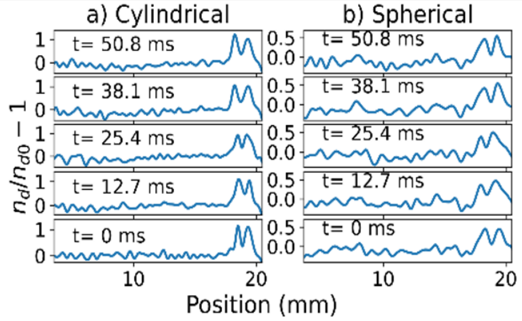


FIG. 2. The time evolution of excited cylindrical and spherical pinned solitons obtained from the simulation studies.

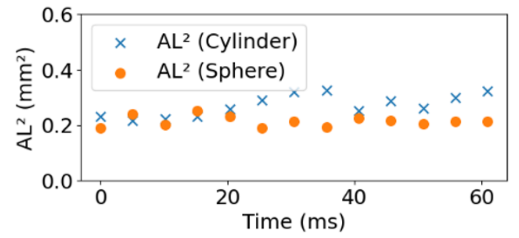


FIG. 3. Variation of soliton parameter with time for cylindrical and spherical pinned solitons

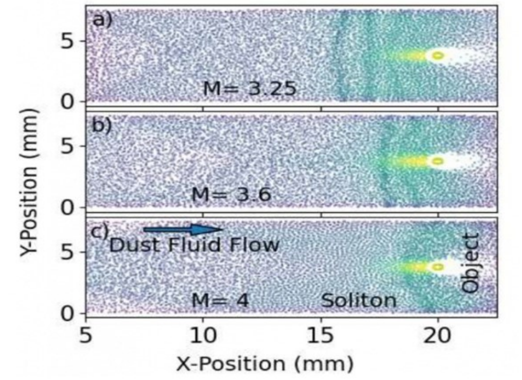


FIG. 4. The excited cylindrical pinned solitons with different supersonic flow velocities of the dust fluid over the cylindrical charged object.

> ***The experiment confirmed the existence of multi-dimensional (cylindrical and spherical) pinned solitons seen in a flowing dusty plasma experiment. These pinned solitons exhibited multi-humped spatial structures with the number of humps decreasing as the flow velocity increased.***

Solitons are a special class of nonlinear localized waves that can propagate for a long time without changing their identity and can retain their shape and size after interacting with each other. These kinds of nonlinear waves arise in a medium through an exact balance between nonlinear steepening and dispersive broadening of the wave. They can be generated by the moving object in a fluid medium (or fluid moving over an object) when its velocity exceeds a critical value, namely, the phase speed of a typical normal mode of the medium. The solitons which are created in front of the object, then move away from it at a speed faster than the speed of the object and are known as precursor solitons. At higher speeds of the object, it is also possible to excite another kind of soliton—one that travels at the same speed as the object and remains attached to it. These are known as pinned solitons. Originally studied in the context of water waves and applications to marine engineering, such solitons have recently become the object of much interest in plasma physics after it was suggested that precursors or pinned in the form of ion-acoustic solitons can be excited by fast moving charged objects in a plasma and may have important applications in the detection of orbital debris objects in the Earth's ionosphere as there is a significant increase of the radar scattering cross section [1]. In a recent work, we have reported the excitation of cylindrical and spherical pinned solitons in a flowing dusty plasma medium by performing the experiments in the Dusty Plasma Experimental (DPEx) device at IPR [2]. The Dusty plasmas are characterized by the Coulomb coupling parameter (Γ), which is the ratio of electrostatic potential energy at mean separation to the average kinetic energy, and the screening parameter (κ), which is the ratio of interparticle distance to Debye length, and it depends on the background plasma parameters. In general, the medium behaves as a solid-like state for Γ greater than 180, and below 180, it shows a fluid-like state. In these experiment of fluid dusty plasmas, It is observed that the nonplanar (cylindrical and spherical) pinned solitons are excited for a particular range of supersonic dust fluid velocity Mach, M~2-4 (Mach number is the ratio of velocity of fluid to the sound speed of sound in that medium) over a charged object of cylindrical (or spherical) shape. An important finding is the multi-humped nature of the spatial structure of these solitons and a decrease in the number of humps with increasing flow velocity.

In order to provide a better understanding of our experimental observations, a number of molecular dynamics (MD) simulations are performed on ANTYA by using the open-source code Large-scale Atomic/ Molecular Massively Parallel Simulator (LAMMPS) [3]. The MD simulations provide the full 6N+1 dimensional phase space of a dynamical system by numerically integrating the equations of motion for each particle. A 3D periodic simulation box is constructed with dimensions lx=150d, ly=50d, and lz=50d (where d represents the inter-particle separation) for 100000 dust charge particles. The parameters of the dust particles are taken to be the same as those of our experiment, namely, mass m_d =8.8×10^{13} kg and charge Q_d=8.23×10^3 e (where e is the electronic charge). The interparticle separation (d) and Debye length (λ_d) are chosen to be 150 mm and 40 mm, respectively. The particles are allowed to interact with each other via the Yukawa interaction potential. In our experiments, the value of Γ lies in the range of 20–100; therefore, we have chosen Γ=45 and κ=3.7 for the present set of MD simulations. The cylindrical (or spherical) shaped charged object is constructed, using an accumulated ensemble of immovable particles, within the simulation box to excite the cylindrical (or spherical) pinned solitons, having the charge on the object is estimated by Q = (V_f - V_p) ×C, where V_f and V_p represent the floating and plasma potential, respectively, and C is the capacitance of a cylindrical or spherical object. Furthermore, the interaction between the charged object and the dust particles is also considered to be of the Yukawa type, and the screening length is taken as 150 mm. A typical wall time of these simulations is around 54 minutes using 80 MPI processors on two nodes of ANTYA. Now, the system is brought into a thermodynamic equilibrium state with the required temperature by distributing and evolving in a canonical ensemble (NVT) using a Nose–Hoover thermostat[4,5]. Then disconnected from the canonical ensemble and allowed to run with a micro-canonical (NVE). The simulation time step is chosen to be 0.001 ω_{pd}^{-1} $(\omega_{pd}^{-1}\ is\ the\ dust\ plasma\ frequency)$ to accurately resolve the dynamics of dust particle response.

After this system reaches to thermodynamic equilibrium state, the dust particles are then made to flow along the x-axis over the cylindrical (or spherical) charged object with a supersonic speed, with M=3.6 (or M=3.55). The supersonic flow is introduced by adding an x-directional velocity on the dust particles. So, there is a rapid accumulation of density in front of the object that cannot disperse quickly enough, as the pileup rate, which is proportional to the flow velocity, is higher than the phase velocity of the dust acoustic speed. This leads to a nonlinear steepening of the density perturbation. The balancing of this steepening by a dispersive broadening leads to the formation of a soliton. Akin to the experimental observation, the solitons get excited in front of the cylindrical (or spherical) charged object as shown in Fig. 1. The top views of these solitons are shown in Figs. 1(a) and 1(b) for cylindrical and spherical charged objects, respectively, whereas their side views are depicted in Figs. 1(c) and 1(d). The overall shapes of the excited wave fronts look like those of the exciting charged object, as also observed in our experiments. The time evolution of amplitude (A) of the multi-humped cylindrical and spherical pinned solitons is then obtained and shown in Fig. 2. This plot indicates that the structures remain stationary in the laboratory frame or in the fluid frame, they move with the same velocity as the object, thereby revealing their pinned nature.

To examine structural properties of solitons, the solitonic parameter, AL^2 (the product of amplitude and square of width), is plotted with time in Fig. 3, which shows that the solitonic parameter remains nearly constant with time. Therefore, Figs. 2 and 3 together imply that these nonlinear excitations are pinned solitons. Then, to investigate the multi-humped solitons for the cylindrical solitons, we simulate three different values of Mach numbers, M= 3.25, 3.6, and 4. 4. Fig. 4. depicts the simulation results, which indicates that the number of humps of the pinned soliton in front of cylindrical charged object reduces as the velocity of the dust fluid flow increases. The molecular dynamics (MD) simulations reproduce the salient features of our experimental results—namely, the creation of pinned solitons over a range of supersonic velocities, the multi-humped nature of their spatial structure, and the decrease in the number of humps with increasing source velocity.

Reference:

1. A Sen, S Tiwari, S Mishra, and P. Kaw, *Adv. Space Res*. 56, 429 (2015).
2. Prasanta Amat, P. Bandyopadhyay, Krishan Kumar, Ajaz Mir, and A. Sen, Phys. Plasmas 32, 073701 (2025).
3. S. Plimpton, J. Comput. Phys 1-19, 093701 (1995).
4. S. Nose, Mol. Phys. 52, 255–268 (1984).
5. W. G. Hoover, Phys. Rev. A 31, 1695 (1985).

Migrating to oneAPI: A Practical Guide for HPC Users

High-Performance Computing (HPC) environments are increasingly embracing **heterogeneous architectures** to meet modern computational demands. Intel's **oneAPI** initiative represents a significant evolution in compiler and toolchain design, aimed at unifying development across CPUs and GPUs.

While many HPC systems still rely on **Intel Classic Compilers** (like intel/2018, intel/2019 intel/2020), transitioning to oneAPI can unlock performance, portability, and maintainability benefits. This article provides a practical guide for HPC users ready to **explore** oneAPI compilers.

1) What is oneAPI?

Intel oneAPI is an open, standards-based programming model targeting multiple architectures. It includes:

- Intel oneAPI DPC++/C++ Compiler
- Intel Fortran Compiler (ifx) for CPU and GPU
- Intel MPI, MKL, and performance libraries, now part of oneAPI
- DPC++ (Data Parallel C++): Based on SYCL and C++17

2) Classic vs. oneAPI Compilers

Feature	Classic (intel/2018, intel/2019)	oneAPI (2021+)
C/C++ Compiler	icc, icpc	icx, dpcpp
Fortran Compiler	ifort	ifx
GPU Offloading	No	Yes (via OpenSYCL)
Standards Support	Older (C++11/14/17)	Modern (C++17/20, SYCL)

3) Preparing the HPC Environment

3.1) Check available modules

```
[user@login1 ~] $ module avail intel
intel-2018      intel18/mkl      intel20/mkl
intel-2019      intel20/icc      intel20/mpi
intel-2020      intel20/impi

[user@login1 ~] $ module avail oneapi
oneapi/modulefiles/advisor/2021.2.0        oneapi/modulefiles/debugger/10.1.1          oneapi/modulefiles/init_opencl/2021.2.0
oneapi/modulefiles/ccl/2021.2.0            oneapi/modulefiles/dev-utilities/2021.2.0   oneapi/modulefiles/inspector/2021.2.0
oneapi/modulefiles/compiler/2021.2.0       oneapi/modulefiles/dnnl/2021.2.0        oneapi/modulefiles/intel_ippcp_intel64/2021.2.0
oneapi/modulefiles/compiler32/2021.2.0     oneapi/modulefiles/dpct/2021.2.0            oneapi/modulefiles/itac/2021.2.0
oneapi/modulefiles/dal/2021.2.0            oneapi/modulefiles/dpl/2021.2.0             oneapi/modulefiles/mkl/2021.2.0
```

3.2) Load oneAPI Compiler

```
[user@login1 ~] $ module load oneapi/modulefiles/compiler/2021.2.0
Loading compiler version 2021.2.0       Loading tbb version 2021.2.0        Loading debugger version 10.1.1
Loading compiler-rt version 2021.2.0    Loading oclfpga version 2021.2.0    Loading init_opencl version 2021.2.0

Loading oneapi/modulefiles/compiler/2021.2.0
  Loading requirement: oneapi/modulefiles/tbb/2021.2.0 oneapi/modulefiles/debugger/10.1.1 oneapi/modulefiles/compiler-rt/2021.2.0
    /home/application/ONEAPI/compiler/2021.2.0/linux/lib/oclfpga/modulefiles/init_opencl oneapi/modulefiles/oclfpga/2021.2.0
```

3.3) *Confirm Compilers*

```
[user@login1 ~] $ which icx
/home/application/ONEAPI/compiler/2021.2.0/linux/bin/icx
[user@login1 ~] $ which icpx
/home/application/ONEAPI/compiler/2021.2.0/linux/bin/icpx
[user@login1 ~] $ which ifx
/home/application/ONEAPI/compiler/2021.2.0/linux/bin/ifx
```

4) Porting Guide: From Classic Intel to oneAPI

4.1) *C/C++ Example*

A) Classic Intel Compiler

```
[user@login1 ~] $ icc -O3 -qopenmp example.c -o example
```

B) oneAPI Equivalent

```
[user@login1 ~] $ icx -O3 -march=native -fopenmp example.c -o example
```

4.2) *Fortran Example*

A) Classic Intel Compiler

```
[user@login1 ~] $ ifort -O3 -qopenmp example.f90 -o example
```

B) oneAPI Equivalent

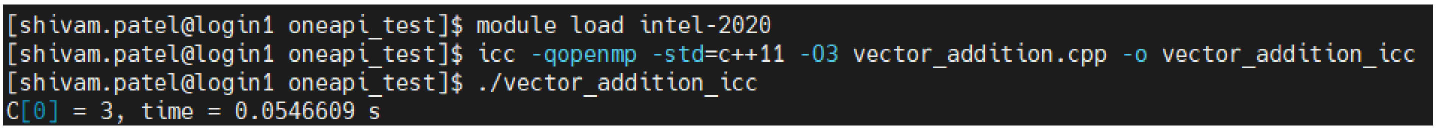
```
[user@login1 ~] $ ifx -O3 -march=native -fopenmp example.f90 -o example
```

5) Key Compatibility Concerns:

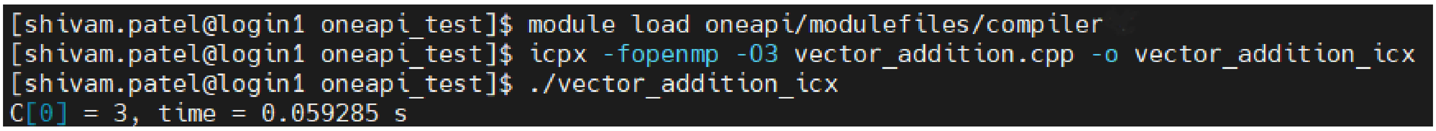
- Compiler Flags Incompatibility (for example, **-qopenmp** changed to **-fopenmp** in oneAPI)
- Precompiled binaries using classic compiler needs to be recompiled to avoid runtime failures.
- Legacy standards might not be supported in latest updates.

6) Example Code to verify working of compiler

A simple vector addition code is taken as an example to check the compilation using both classical intel compiler and oneAPI based intel compiler.



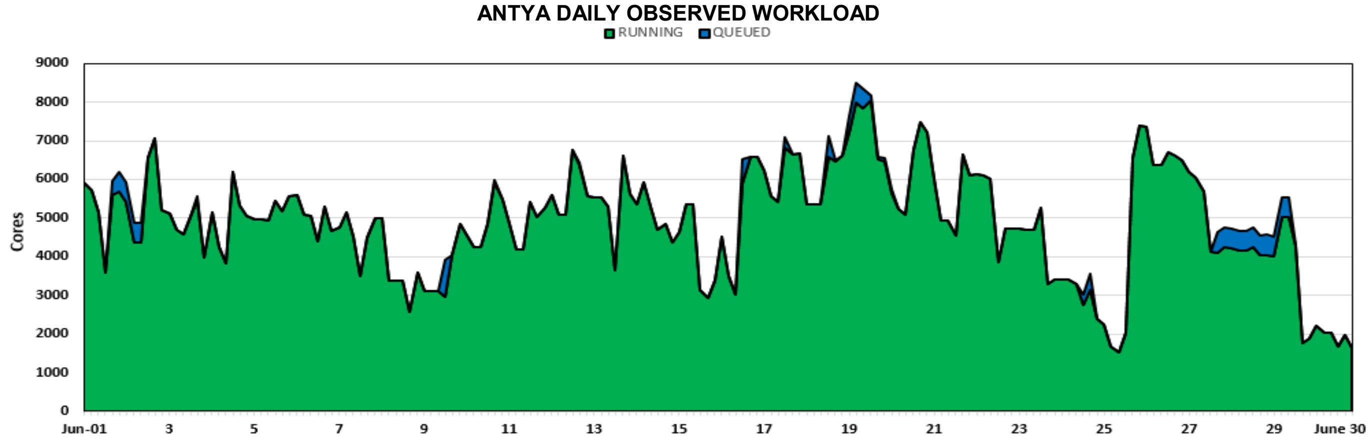
```
[shivam.patel@login1 oneapi_test]$ module load intel-2020
[shivam.patel@login1 oneapi_test]$ icc -qopenmp -std=c++11 -O3 vector_addition.cpp -o vector_addition_icc
[shivam.patel@login1 oneapi_test]$ ./vector_addition_icc
C[0] = 3, time = 0.0546609 s
```



```
[shivam.patel@login1 oneapi_test]$ module load oneapi/modulefiles/compiler
[shivam.patel@login1 oneapi_test]$ icpx -fopenmp -O3 vector_addition.cpp -o vector_addition_icx
[shivam.patel@login1 oneapi_test]$ ./vector_addition_icx
C[0] = 3, time = 0.059285 s
```

Migrating from Intel 2018 or 2019 (Classic) to OneAPI introduces several compatibility considerations, including compiler binary changes, deprecated flags, stricter language standards, and potential differences in OpenMP behavior. While OneAPI offers modern optimizations and support for newer hardware, careful testing is required to ensure correctness and performance. By updating build scripts, replacing deprecated flags, and verifying outputs, users can transition smoothly while leveraging OneAPI's advanced features for future HPC development. For legacy applications, retaining the classic compilers may be necessary, but adopting OneAPI ensures long-term compatibility with Intel's evolving toolchain.

ANTYA Utilization: JUNE 2025



ANTYA HPC Users' Statistics

June 2025

Total Successful Jobs~ 2273

♦ **Top Users** (Cumulative Resources)

CPU Cores	**Amit Singh**
Walltime	**Amit Singh**
Jobs	**Jugal Chowdhury**
GPU Cards	**Shishir Biswas**

ANTYA Usage, Updates and News

- **Scheduled Downtime**: There was no downtime of ANTYA for June 2025.
- **Job Submissions**: The highest job loads were observed in the *regularq*, mediumq, serialq and *longq* queues, reflecting sustained user activity across multiple workloads in various queues.
- **Cluster Utilization**: The system maintained an average utilization of approximately 48.81% and peak utilisation of 80.57%.

Packages/Applications Installed: WarpX
(advanced **electromagnetic & electrostatic Particle-In-Cell** code)
$ module load warpx

Other Recent Work on HPC

Development of Water-Cooled Half Helix Antenna for Ion Cyclotron Resonance Heating (ICRH) in Plasma Thruster Experiments	**Nitesh Kataria**
Microwave-Induced Plasma in Waveguide-Fed Cylindrical Cavity: Design and Analysis using Computational Simulations	**Dr. Pratik Ghosh**
Parallel Computing with MATLAB 2023a: Job Submission via MATLAB Parallel Server Integrated with PBS Scheduler on ANTYA HPC Cluster	**Arvind Mohan Singh**

Acknowledgement

The HPC Team, Computer Division IPR, would like to thank all Contributors for the current issue of *GAṆANAM*.

Join the HPC Users Community

hpcusers@ipr.res.in

If you wish to contribute an article in

GAṆANAM (गणनम्)

Contact us

HPC Team

Computer Division, IPR

Disclaimer: "*GAṆANAM*" is IPR's informal HPC Newsletter to disseminate technical HPC related work performed at IPR from time to time. Responsibility for the correctness of the Scientific Contents including the statements and cited resources lies solely with the Contributors.