# Numerical Simulation

***Counsel:*** **Tuesday & Thursday 11:30 - 13:00**

***Office:*** **Room 532**

***Homepage:*** **//solardynamicslab.khu.ac.kr/~magara**

***Goal:***

- **Understand basic features of partial differential equations (PDEs) used in physics**
- **Understand numerical schemes of simulations**
- **Use these schemes to convert PDEs to the forms suitable for performing simulations**
- **Acquire basic skills of writing numerical codes and analyzing simulation data**

***Lecture characteristics:***

**Theory: 30%, Experiments/Hands-on Practice: 40%**
**Practical Training: 30%**

***Instruction method:***

**Lecture, Discussion, Audi-visual Education, Practice**

***Evaluation method:***

**Mid-term Exam... 30%, Final Exam... 30%, Homework/Report... 30%,**
**Attendance... 10%**

## *Textbooks:*

- Computational Techniques for Fluid Dynamics 1: Fundamental and General Techniques (Clive A.J. Fletcher, Springer, 1991, 9783540530589)

- Computational Techniques for Fluid Dynamics 2: Specific Techniques for Different Flow Categories (Clive A.J. Fletcher, Springer, 1991, 9783540536017)

- Riemann Solvers And Numerical Methods for Fluid Dynamics: A Practical Introduction (Eleuterio F. Toro, Springer-Verla, 2009, 9783540252023)

## *Assignments:*

- Write numerical codes in programming languages
- Run these codes to perform simulations & analyze simulation data

# What is numerical simulation?

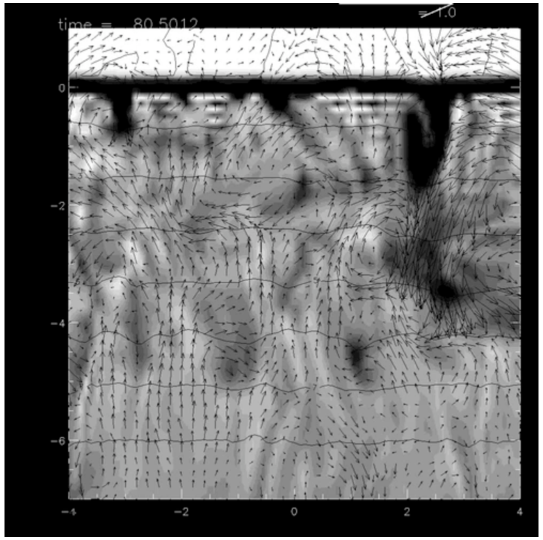
***Three key features of numerical simulation discussed in this lecture...***

1. Numerical simulation **reproduces physical phenomena evolving** in **time** and **space**.

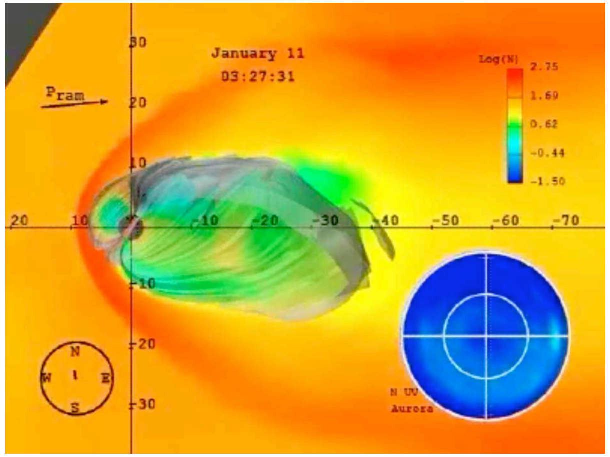
2. Numerical simulation **is performed** in a **computer** according to a **numerical code** written in a **programing language** (fortran, c, etc.).

3. Numerical simulation **is based** on **partial differential equations (PDEs)** that **describe temporal** & **spatial variations** of **physical quantities** such as density, velocity, pressure, and magnetic field.

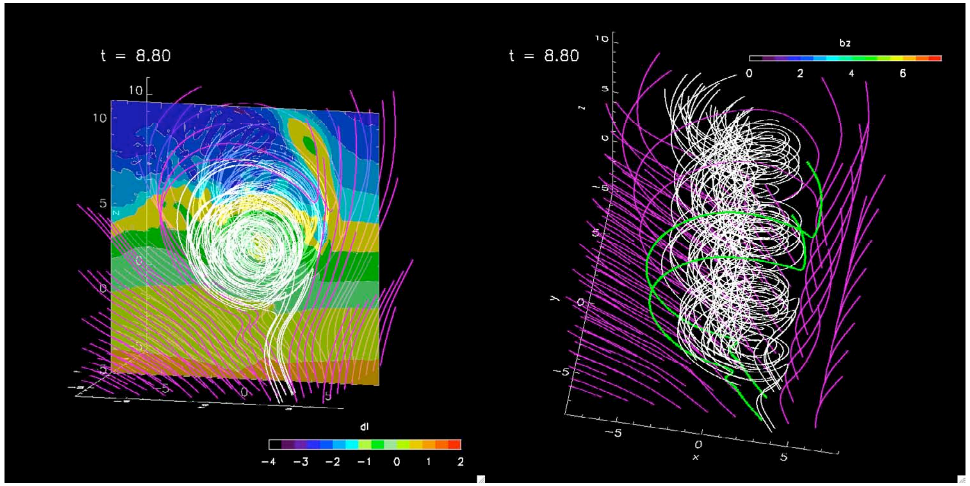
## 1. Numerical simulation *reproduces physical phenomena evolving* in *time* and *space*.




convective motion in a solar interior

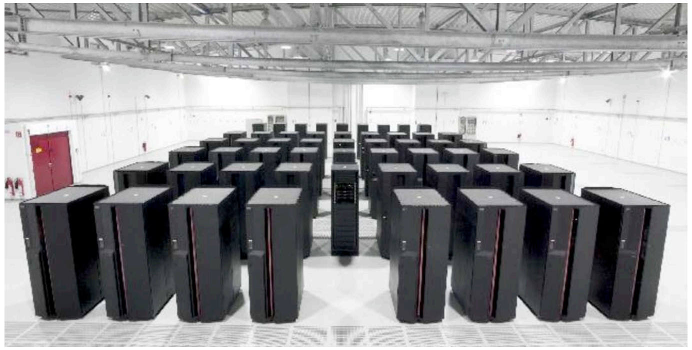



solar wind interacting with magnetosphere




solar penumbral microjet

## 2. Numerical simulation *is performed* in a *computer* according to a *numerical code* written in a *programing language (fortran, c,* etc.*)*.



(super)computer conducts calculatior according to the code

submit a code to a computer

```fortran
      hwa4 =        2.*(ah(i+1,j,k,4) -ah(i-1,j,k,4))
&          + uz1(j)*uy1(k)*(yah(i,j,k,4)  -yah(i-1,j,k,4))
&          + uz2(j)*uy1(k)*(yah(i,j-1,k,4)-yah(i-1,j-1,k,4))
&          + uy2(k)*uz1(j)*(yah(i,j,k-1,4)-yah(i-1,j,k-1,4))
&          + uz2(j)*uy2(k)*(yah(i,j-1,k-1,4)-yah(i-1,j-1,k-1,4))

      hwa5 =        2.*(ah(i+1,j,k,5) -ah(i-1,j,k,5))
&          + uz1(j)*uy1(k)*(yah(i,j,k,5)  -yah(i-1,j,k,5))
&          + uz2(j)*uy1(k)*(yah(i,j-1,k,5)-yah(i-1,j-1,k,5))
&          + uy2(k)*uz1(j)*(yah(i,j,k-1,5)-yah(i-1,j,k-1,5))
&          + uz2(j)*uy2(k)*(yah(i,j-1,k-1,5)-yah(i-1,j-1,k-1,5))

      hwa6 =        2.*(ah(i+1,j,k,6) -ah(i-1,j,k,6))
&          + uz1(j)*uy1(k)*(yah(i,j,k,6)  -yah(i-1,j,k,6))
&          + uz2(j)*uy1(k)*(yah(i,j-1,k,6)-yah(i-1,j-1,k,6))
&          + uy2(k)*uz1(j)*(yah(i,j,k-1,6)-yah(i-1,j,k-1,6))
&          + uz2(j)*uy2(k)*(yah(i,j-1,k-1,6)-yah(i-1,j-1,k-1,6))

      hwa7 =        2.*(ah(i+1,j,k,7) -ah(i-1,j,k,7))
&          + uz1(j)*uy1(k)*(yah(i,j,k,7)  -yah(i-1,j,k,7))
&          + uz2(j)*uy1(k)*(yah(i,j-1,k,7)-yah(i-1,j-1,k,7))
&          + uy2(k)*uz1(j)*(yah(i,j,k-1,7)-yah(i-1,j,k-1,7))
&          + uz2(j)*uy2(k)*(yah(i,j-1,k-1,7)-yah(i-1,j-1,k-1,7))

      hwa8 =        2.*(ah(i+1,j,k,8) -ah(i-1,j,k,8))
&          + uz1(j)*uy1(k)*(yah(i,j,k,8)  -yah(i-1,j,k,8))
&          + uz2(j)*uy1(k)*(yah(i,j-1,k,8)-yah(i-1,j-1,k,8))
&          + uy2(k)*uz1(j)*(yah(i,j,k-1,8)-yah(i-1,j,k-1,8))
&          + uz2(j)*uy2(k)*(yah(i,j-1,k-1,8)-yah(i-1,j-1,k-1,8))


      gwa1 =        2.*(ag(i,j+1,k,1) -ag(i,j-1,k,1))
&          + ur1(i)*uy1(k)*(yag(i,j,k,1)  -yag(i,j-1,k,1))
&          + ur2(i)*uy1(k)*(yag(i-1,j,k,1)-yag(i-1,j-1,k,1))
&          + ur1(i)*uy2(k)*(yag(i,j,k-1,1)-yag(i,j-1,k-1,1))
&          + ur2(i)*uy2(k)*(yag(i-1,j,k-1,1)-yag(i-1,j-1,k-1,1))

      gwa2 =        2.*(ag(i,j+1,k,2) -ag(i,j-1,k,2))
&          + ur1(i)*uy1(k)*(yag(i,j,k,2)  -yag(i,j-1,k,2))
&          + ur2(i)*uy1(k)*(yag(i-1,j,k,2)-yag(i-1,j-1,k,2))
&          + ur1(i)*uy2(k)*(yag(i,j,k-1,2)-yag(i,j-1,k-1,2))
&          + ur2(i)*uy2(k)*(yag(i-1,j,k-1,2)-yag(i-1,j-1,k-1,2))
```

1919 6

numerical code written in *fortran*

# Why fortran?

Among various programing languages *fortran* (formula translation) is relatively old.
Then why do we use it for numerical simulation?

In fact, *fortran* is like the train, compared to other modern programing languages which are like the car.

## *fortran*

**Long history** (train: since 16th century)

Disadvantages: **less flexible, less controllable, less interactive**

Advantage: **very fast (arithmetic operations)**

↓

numerical simulation, any program based on arithmetic operations

## *c, c++, Python, etc.*

**Short history** (car: since 18th century)

Advantages: **more flexible, more controllable, more interactive**

Disadvantage: **relatively slow**

↓

data analysis, instrument control, application program, OS