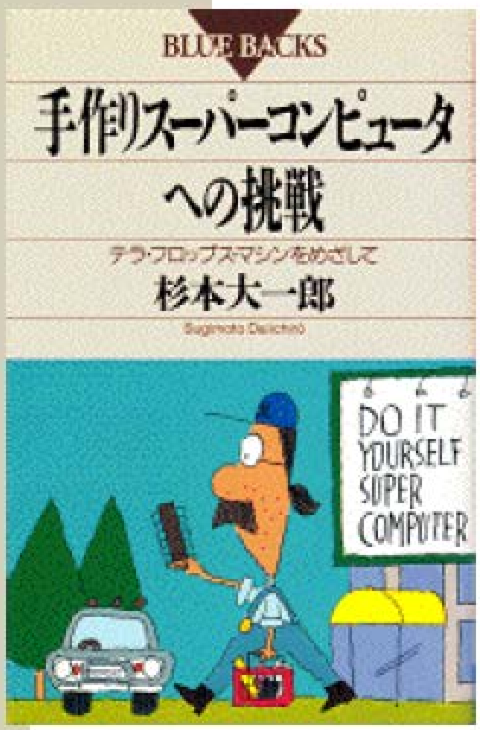
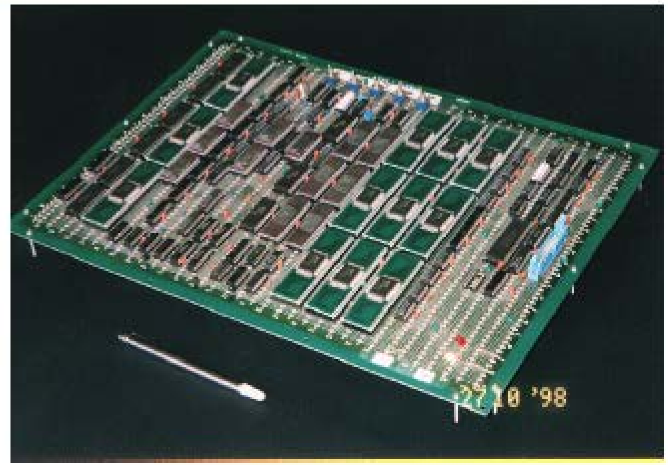
# GPUによる反水素生成
# --CUNBODYとSIMBUCA--

飯高敏晃（理研戎崎計算宇宙,iTHES）
Simon van Gorp(理研山崎原子物理)

# （I） 演算加速器研究

# 初代GRAPEは20万円



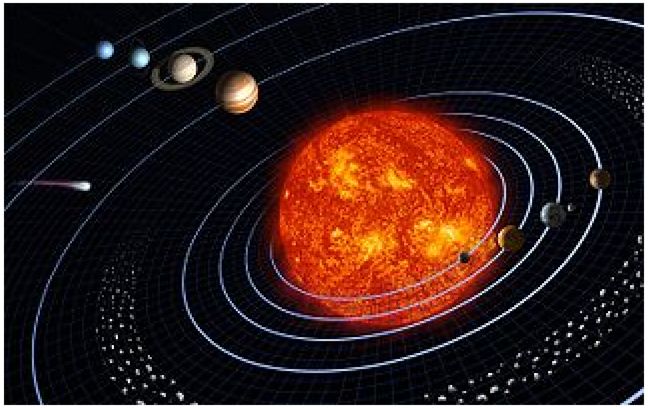



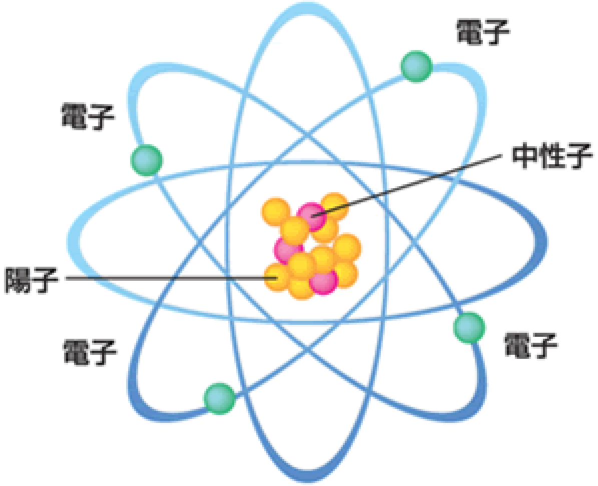
なぜ安かったか

秋葉原で市販品（量産品）の部品を買ってきて組み立てた。

http://www.iitaka.org/bookshelf_j.html#special_computer

# 宇宙⇔原子　対称性





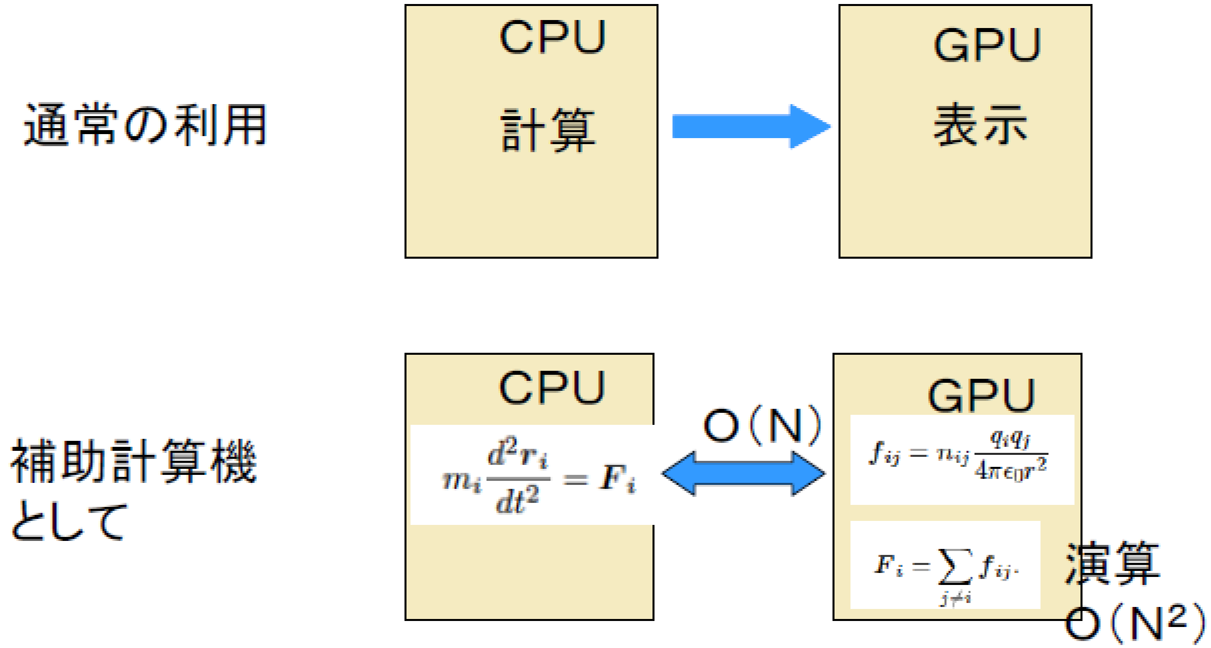

$$F_i = G\sum_j m_i m_j \frac{(\vec{r}_j - \vec{r}_i)}{|\vec{r}_j - \vec{r}_i|^3}$$

重力

$$F_i = -k\sum_j q_i q_j \frac{(\vec{r}_j - \vec{r}_i)}{|\vec{r}_j - \vec{r}_i|^3}$$

クーロン力

# CPUとGPU


通常の利用
CPU
計算
GPU
表示
補助計算機として
CPU
$m_i \frac{d^2 \boldsymbol{r}_i}{dt^2} = \boldsymbol{F}_i$
O（N）
GPU
$f_{ij} = n_{ij} \frac{q_i q_j}{4\pi\epsilon_0 r^2}$
$F_i = \sum_{j \neq i} f_{ij}.$
演算
O（N2）

# Cg言語(GPU用C言語)

クーロン力の和の計算

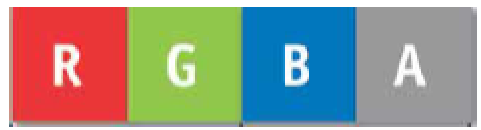
$$f_{ij} = n_{ij} \frac{q_i q_j}{4\pi\epsilon_0 r^2}$$

```
struct Output {
  float4 color : COLOR;
};

Output main(
                   float2 i : TEXCOORD0,
                   uniform int mx,
                   uniform int my,
                   uniform samplerRECT terture):COLOR
{
  Output OUT;
  float r,rr,k,l;
  float2 j;
  float3 ri,rj,rij,force;
  const float eps2=1e-6;
  force=0;
  ri = texRECT(texture,i);
for(l=0.5; l <my; ++l){
for(k=0.5; k <mx; ++k){
    j=float2(k,l);
      rj = texRECT(texture,j);
      rij= rj-ri;
      rr = eps2+dot(rij,rij);
      r  = sqrt(rr);
      force += rij/(r*rr) ;
   };
  };
  OUT.color.xyz=force;
  return OUT;
}
```

# GPUでの数値表現

- 各色が単精度浮動小数点(32bit)に対応
- 画素の構成




- 例：128x128粒子の座標

# MD-GRAPEとの比較

| Data | GF7800GTX | MDGRAPE-2[3] | MDGRAPE-3[4] | |
|---|---|---|---|---|
| # of pipeline | 32 | 4 | 20 | 20 |
| clock (MHz) | 400 | 100 | 460 | 250 |
| peak (Gflops) | 150 | 16.4 | 300 | 165 |
| peak (Gpair/sec) | 6 | 0.4 | 9 | 5 |
| sustained (Gflops) | 15 | 3.75 | NA | (165) |
| sustained (Gpair/sec) | 0.6 | 0.09 | NA | (5) |


- [3] R. Susukita et al., Phys. Commun. 155, 115 (2003).
- [4] M.Taiji et al., in Proceedings of SC'03, November 15-21, 2003, Phoenix Arizona, USA.


http://mdgrape.gsc.riken.jp/?easiestml_lang=xlang:ja

# Why a GPU?

GPU

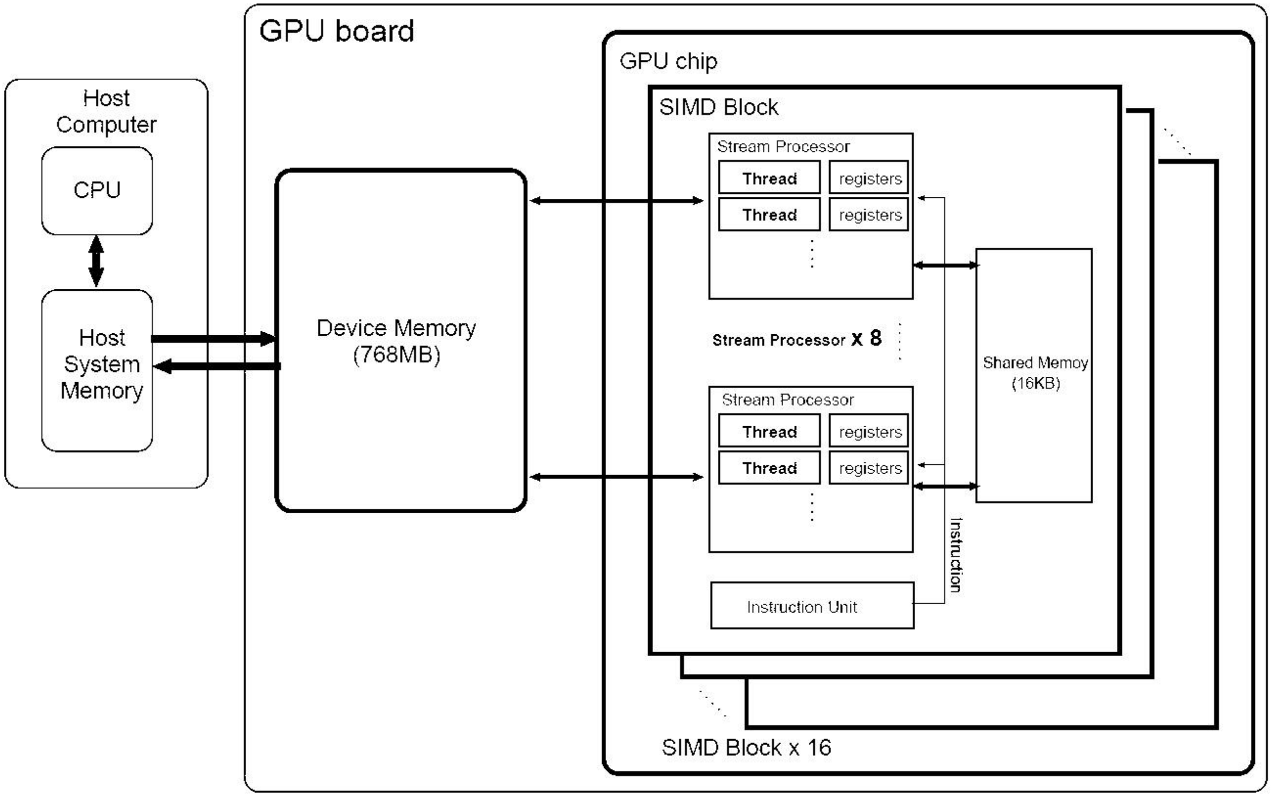
-high parallelism (pipelining!)

-very fast floating point calculations

-CUDA programming

language for GPU's

Geforce 8800 GTX



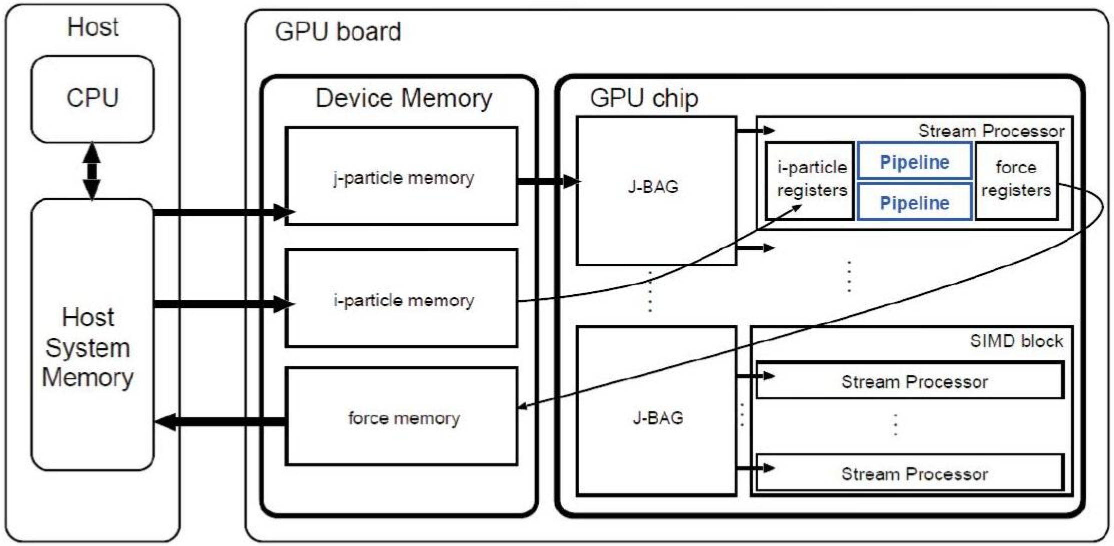

Stream processor

≈ CPU

= Comparable with a factory assembly line with threads being the workers

# Chamomile scheme

- Calculating gravitational interactions on a Graphics Card via the Chamomile scheme from Hamada and Iitaka (in 2007).




i-particles piece available for each 'assembly line'

j-particles piece presents itself sequentially to each line

force is the output of each line


[7]: T. Hamada and T. Iitaka, *arXiv.org:astro-ph/0703100, 2007*
http://arxiv.org/pdf/astro-ph/0703100v1.pdf

# Chamomile scheme: practical usage

- Function provided by Hamada and Iitaka:

cunbody1_force(xj, mj, xi, eps, ai, nmax, nmax)

- Gravitational force ≈ Coulomb Force

$$\mathbf{F}_{grav} = G\frac{Mm}{|\mathbf{r}|^2}\hat{\mathbf{r}} \qquad \mathbf{F}_{coulomb} = k_e\frac{Qq}{|\mathbf{r}|^2}\hat{\mathbf{r}}$$

- Conversion coefficient:

$$a_{Coulomb} = -\frac{q^2 k_e}{m} ai;$$

Needed: - 64 bit linux

- NVIDIA Graphics Card that supports CUDA
- CUDA environment v2.3 - 5.0

Not needed: -CUDA knowledge

-...

# 反水素

# 背景

- 宇宙誕生時は通常の物質と同量あったはずの反物質は、現在の宇宙には同量ない。理由を解明すれば、宇宙誕生の謎に迫れる。

# CPT対称性

- パリティ(P)変換

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \xrightarrow{P} \begin{pmatrix} -x \\ -y \\ -z \end{pmatrix}$$

- 電荷(C)変換　　Q→ーQ
- 時間(T)変換　　T→　ーT
- 普通の古典力学、量子力学、場の理論はC、P,Tそれぞれについて対称(不変)
- 例:

$$m_1 \frac{d^2 r_1}{dt^2} = \frac{Q_1 Q_2}{4\pi\varepsilon_0} \frac{r_1 - r_2}{\left| r_1 - r_2 \right|^{3/2}}$$

$$m_2 \frac{d^2 r_2}{dt^2} = \frac{Q_2 Q_1}{4\pi\varepsilon_0} \frac{r_2 - r_1}{\left| r_2 - r_1 \right|^{3/2}}$$

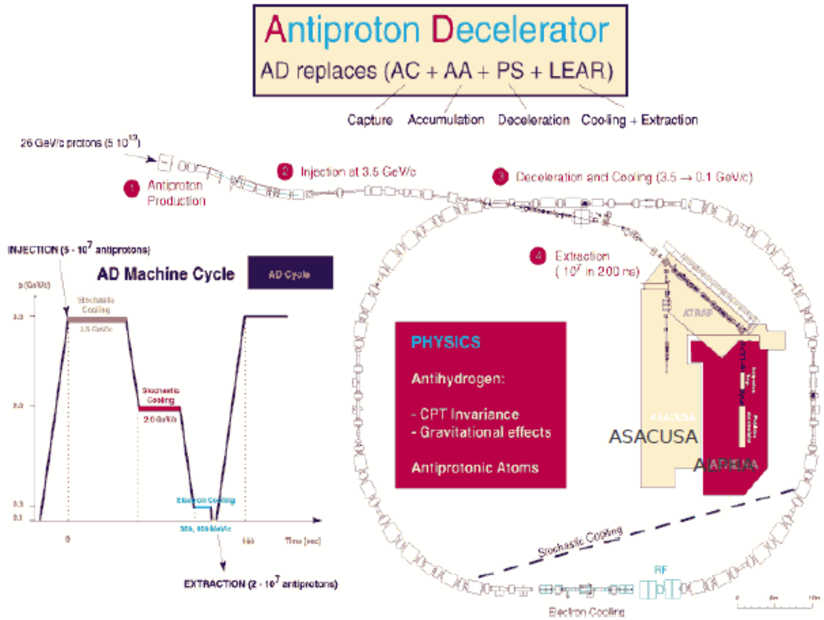
古典水素原子⇔古典反水素原子

# CPT対称性の破れ

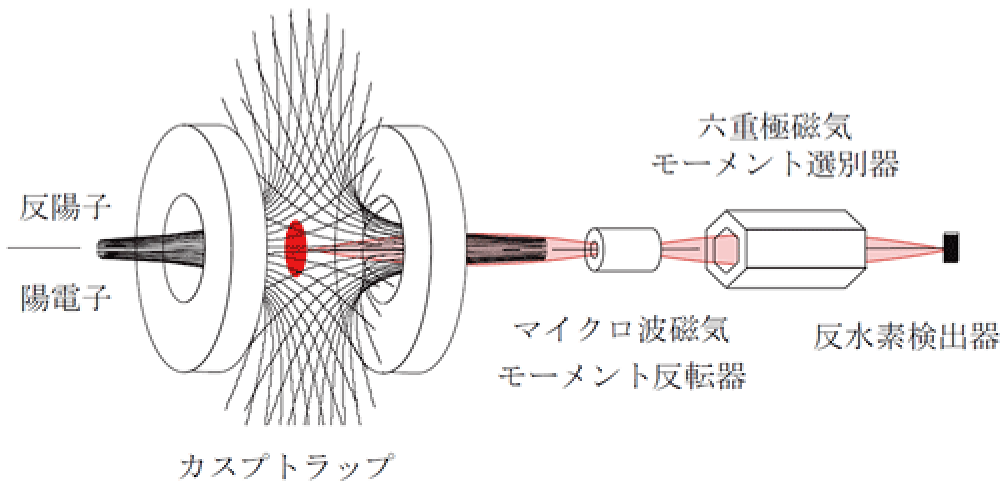
- C,P,CP対称性の破れは見つかった。
- CPT対称性の破れは見つかっていない。
- 普通の「場の量子論」では破られない（証明済）
- ほんとうにそうか。実際はどうか。水素と反水素のエネルギーレベルを比較
- 低エネルギー反水素を生成する必要

# Ion Trap

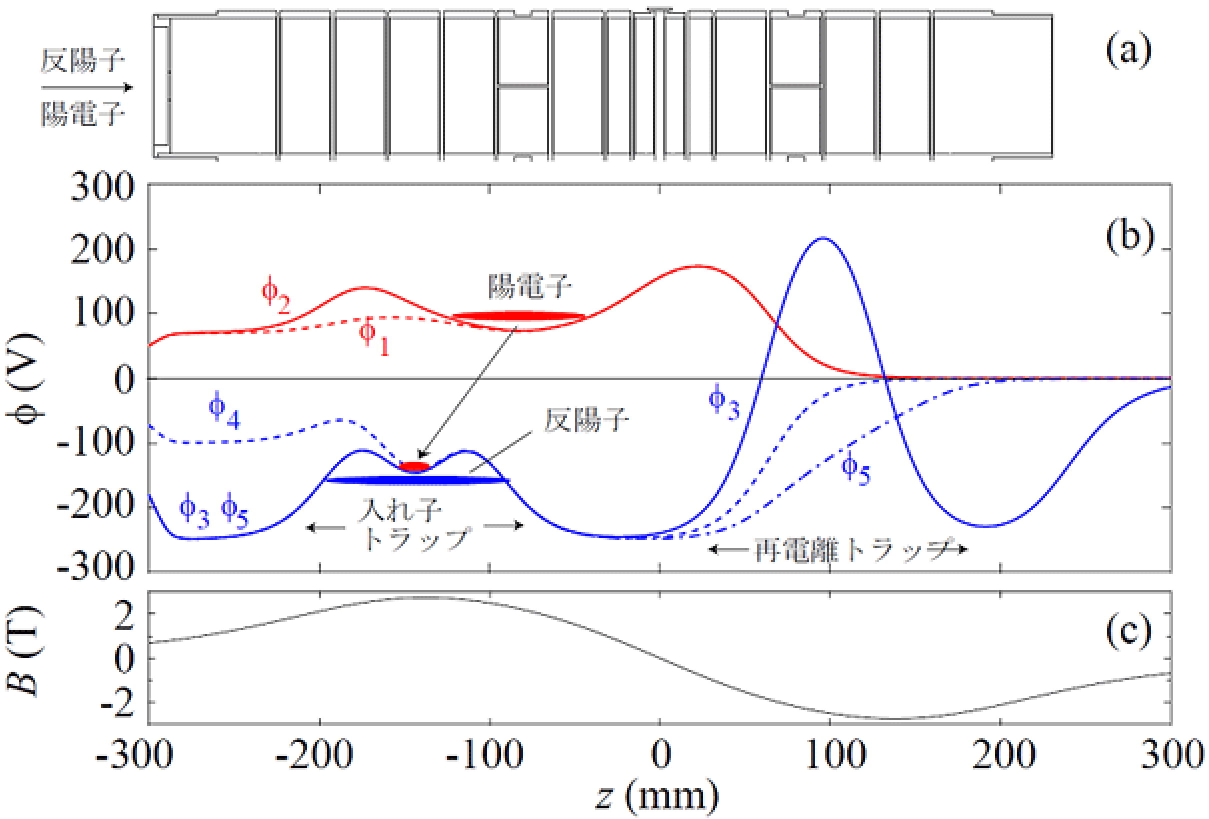
- 反陽子は、真空から対生成によって作られる。～GeVのエネルギー→冷却する必要がある。




http://athena-positrons.web.cern.ch/ATHENA-positrons/wwwathena/graphics/AD_OVERVIEW.gif




反陽子とポジトロンをIon Trap に捕まえて反水素を合成、測定
http://www.riken.go.jp/pr/press/2010/20101206_2/digest/
http://www.riken.go.jp/en/pr/press/2010/20101206_2/




反水素合成には**クーロン相互作用をする**反陽子とポジトロンの雲の運動をうまく制御する必要

⇒Ion Trap Simulation

http://www.riken.go.jp/en/pr/press/2010/20101206_2/

# SIMBUCA:

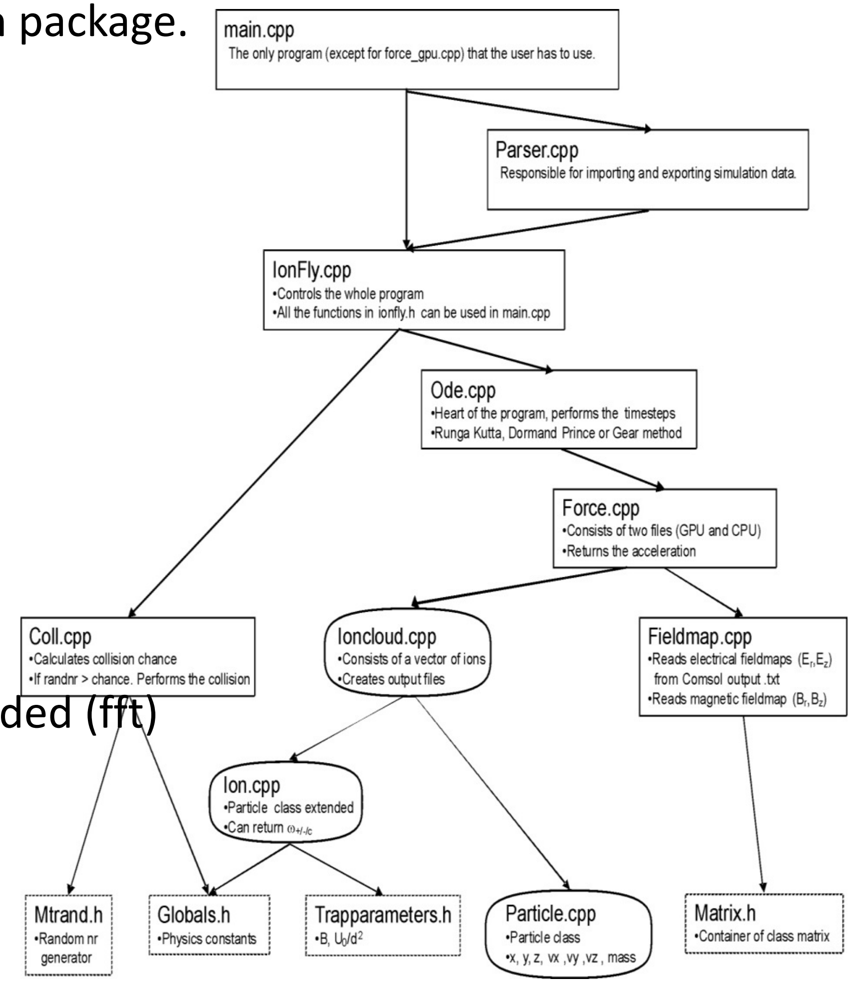
Simon van Gorp

## Ion Trap Simulation

# Simbuca overview

Simbuca is a modular Penning trap simulation package.

- Reading external fieldmaps
  - Comsol
  - SIMION
  - ….
- Trap excitations
  - 3 different integrators
  - 3 buffer gas routines
  - Can run on CPU and GPU
  - Compile with g++ or icpc
  - Several analysis tools are provided (fft)
  - A Makefile is provided
  - Support by me ☺
- http://sourceforge.net/p/simbuca/wiki/





[5]: T. Hamada and T. Iitaka (2007), *arXiv.org:astro-ph/0703100*
http://arxiv.org/pdf/astro-ph/0703100v1.pdf

# Why ?

Understanding the effect of Coulomb interaction where calculations are not possible
Initially created for the WITCH experiment (Energy and position of the ion cloud or $^{35}Ar^{1+}$).

## How ?

Using a timestep integrator (f.e. Runge Kutta adaptive 4$^{th}$ or Gear or Dormand-Price 5$^{th}$ order) to calculate the particle tracks due to the presence of Electric and Magnetic fields. Initially written for Penning traps – now also used for Paul traps, RFQs, MR-TOFs, …

Coulomb interaction is calculated on a Graphics card (GPU) which is much faster than the conventional CPU (see later).

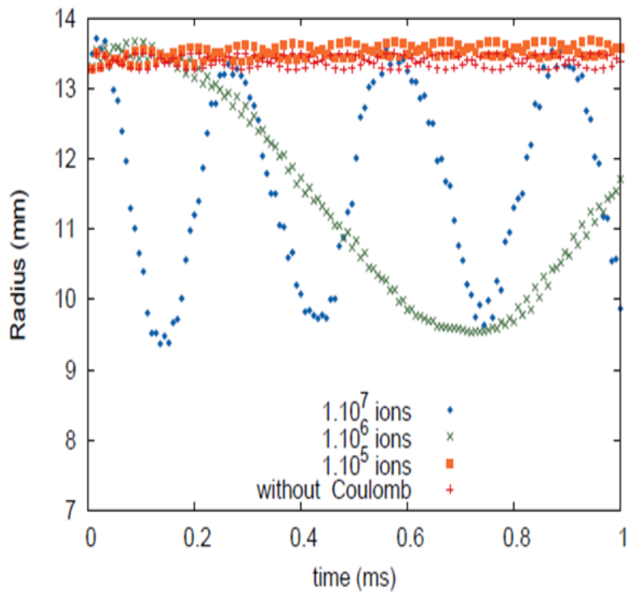
## Example

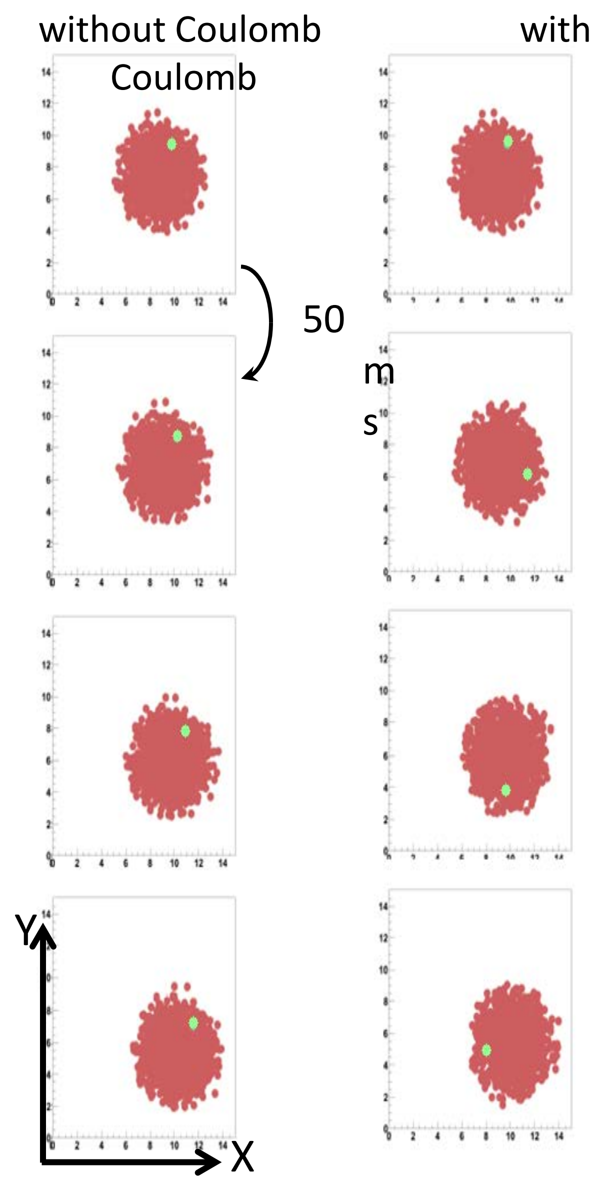
Test of Simbuca by comparing it with the equations of motion of a dipolar excitation at the eigenfrequency . Perfect correspondence between Simulation and theory is found [1].

[1]: S. Van Gorp, et al. (2011), *Nuclear Instruments and Methods in Physics research A, 638, 192-200*

# Other space charge effects examples:

When trapping a large amount of $^{133}Cs^{+}$, the cloud's own electric field will create an E x B drift force for the ions which scales with particle number



# まとめ

- CPT対称性の破れと反水素
- クーロン相互作用をする反陽子･ポジトロンの雲の操作と反水素生成のシミュレーション
- SIMBUCAパッケージ（Ion Trap Simulation）
- CUNBODYライブラリ（クーロン相互作用）