スーパーコンピューティングへの招待

スパコンへの数学・スパコンへの算数

「行列・ベクトル」を勉強しよう！

中島 研吾

東京大学情報基盤センター

東京大学
THE UNIVERSITY OF TOKYO

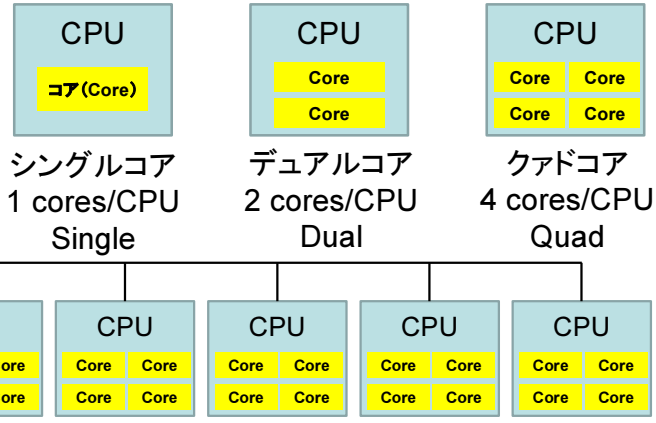
東京大学情報基盤センター
INFORMATION TECHNOLOGY CENTER, THE UNIVERSITY OF TOKYO

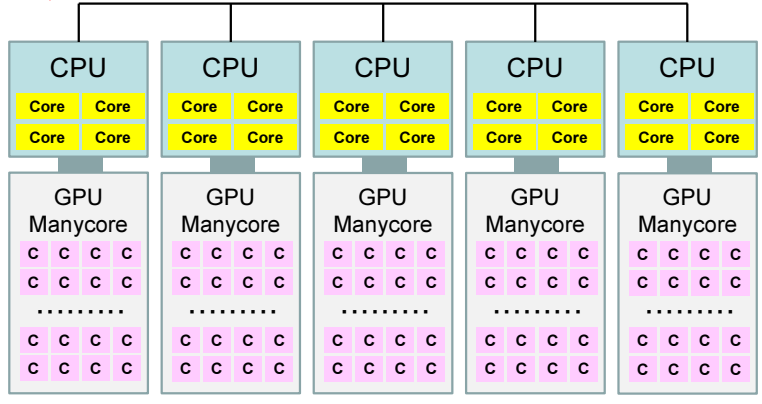
2025年10月25日

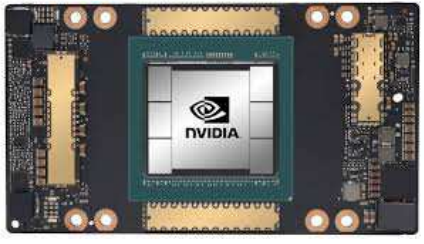
- スーパーコンピュータとは
- 第3の科学：計算科学
- シミュレーションと連立一次方程式
- 行列とベクトル
- 数値線形代数とその実用例

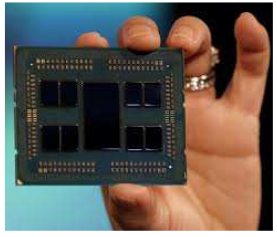
スーパーコンピュータ(スパコン)とは？

- CPU(Central Processing Unit, 中央演算装置)
  - 21世紀以降:マルチコア化，メニィコア化:省電力
  - 並列処理，並列計算:PC，スマホも同様
- GPU(Graphic Processing Unit)
  - 描画用途⇒高いデータ転送能力を計算に利用
  - 数千～数万のコアを有する「メニィコア」
- **マルチコア・メニィコアCPU(+GPU)を高速ネットワークで接続した並列計算機(Parallel Computer)が「スパコン」**





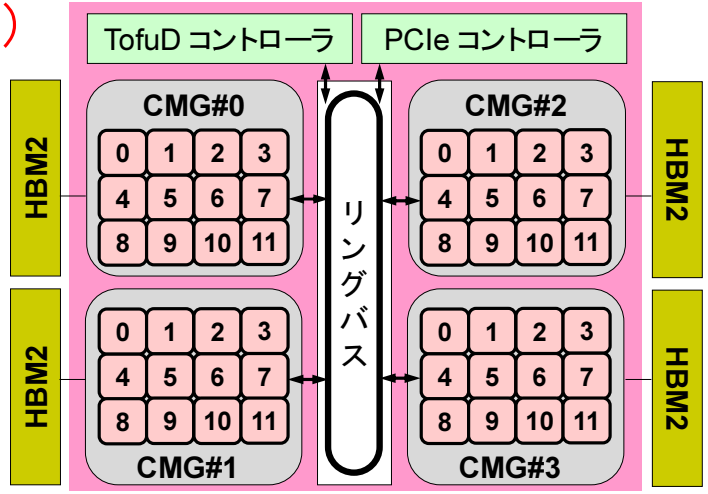




スーパーコンピュータ(スパコン)の性能

- FLOPS値(Floating Point Operations per Second, 浮動小数点演算)
  - 1秒当たりの(倍精度)実数演算性能
  - 10^6 FLOPS= 1 Mega FLOPS = 1 MFLOPS (百万)
  - 10^9 FLOPS= 1 Giga FLOPS = 1 GFLOPS (十億)
  - 10^{12} FLOPS= 1 Tera FLOPS = 1 TFLOPS (兆)
  - 10^{15} FLOPS= 1 Peta FLOPS = 1 PFLOPS (千兆) 「京」: 10 PFLOPS
  - 10^{18} FLOPS= 1 Exa FLOPS = 1 EFLOPS (百京) 富岳: 442 PFLOPS

- 東大センターの「Wisteria/BDEC-01(Odyssey)(シミュレーションノード群)」(富岳と同じ)
  - Fujitsu/Arm A64FX, 1コア性能は70.4GFLOPS
    - 1秒間に704億回の倍精度実数演算
  - 1CPU(1ノード):48コア
    - 3,379.2 GFLOPS= 3.379 TFLOPS
      - 3兆3,792億回
  - 全システム:7,680 CPU(ノード)
    - 25.95 PFLOPS: 2京5,952兆2,600億回



- Wisteria/BDEC-01は，「富岳」と同じCPUを搭載した「「Odyssey」とGPUを搭載した「Aquarius」の2つの部分から構成されています。

- Odyssey
  - 最大処理能力は26PFLOPS
  - 1秒間に，2京6,000兆回の実数演算を実行可能
  - 世界で73位のシステム
- Aquarius
  - 7.27PFLOPS
  - 7,270兆回
  - 262位です

Odyssey
富岳と同じCPU
26PFLOPS，73位

Aquarius
GPU
7.27PFLOPS，262位

スーパーコンピュータは何をするのか？：大量の計算

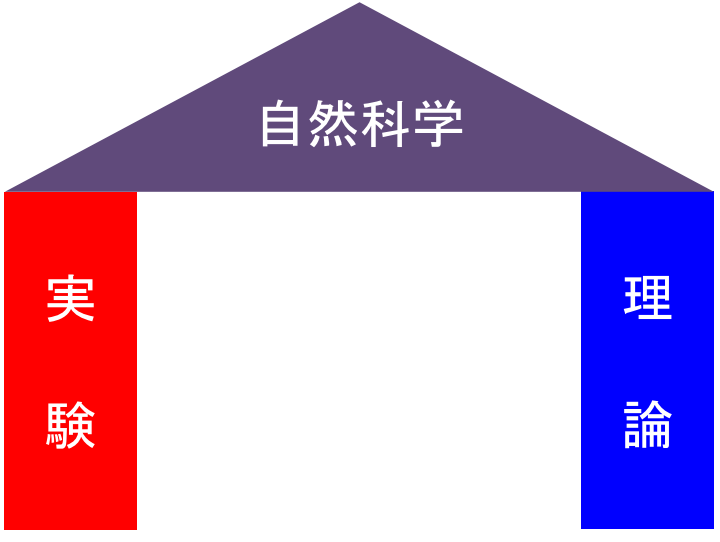
- Odyssey　26 PFLOPS：　2京5,952兆回（世界73位）
- Miyabi-G　73 PFLOPS：　7京2,800兆回（世界37位）

- 富岳　537 PFLOPS：　53京7,000兆回（世界7位，アジア1位）
- El Captain　2,746 PFLOPS：274京6,000兆回（世界1位）（=2.746 EFLOPS）

- スーパーコンピュータとは
- **第3の科学:計算科学**
- シミュレーションと連立一次方程式
- 行列とベクトル
- 数値線形代数とその実用例

自然科学：実験＋理論

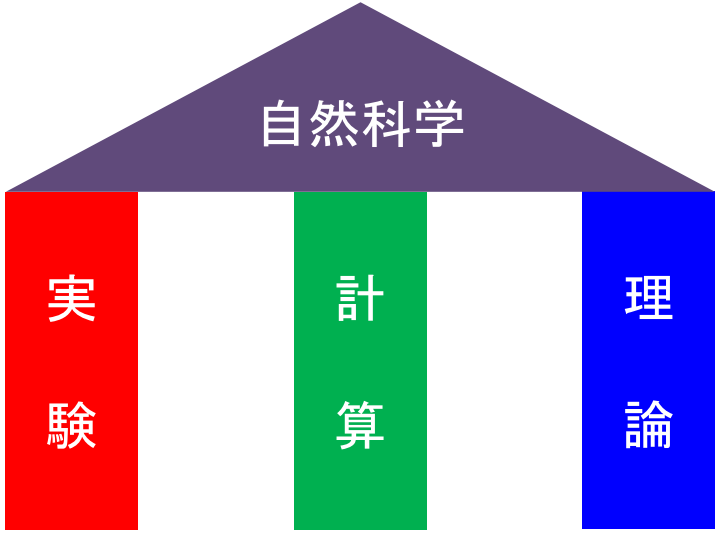
自然科学：「実験（Experiment）」と「理論（Theory）」 ⇒ 2本の柱



第3の科学：計算科学（1990年代）

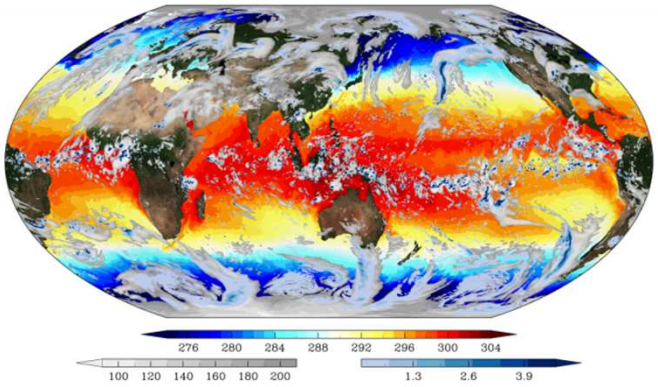
スーパーコンピュータ（スパコン）によるシミュレーションに基づく「計算科学」

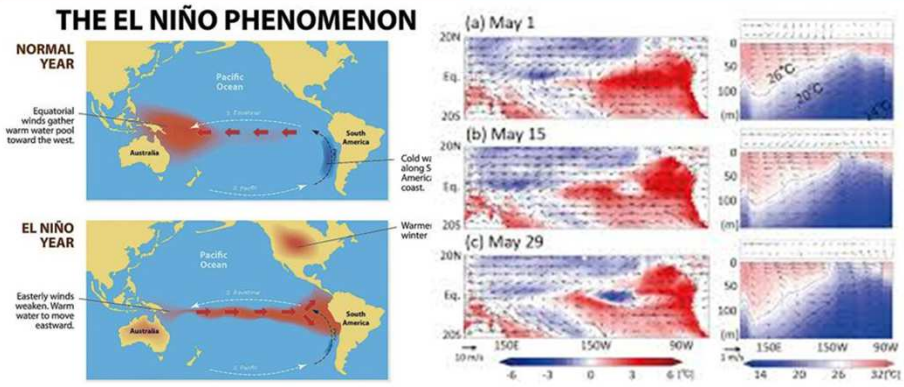
「第3の科学」，「科学の3本目の柱」

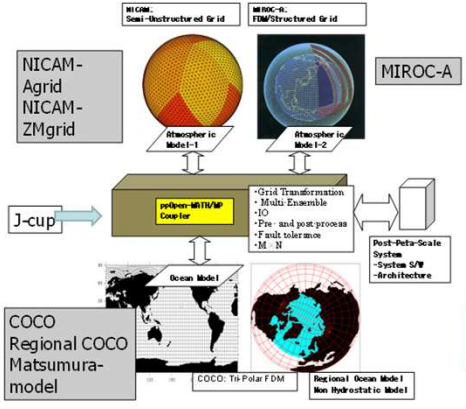


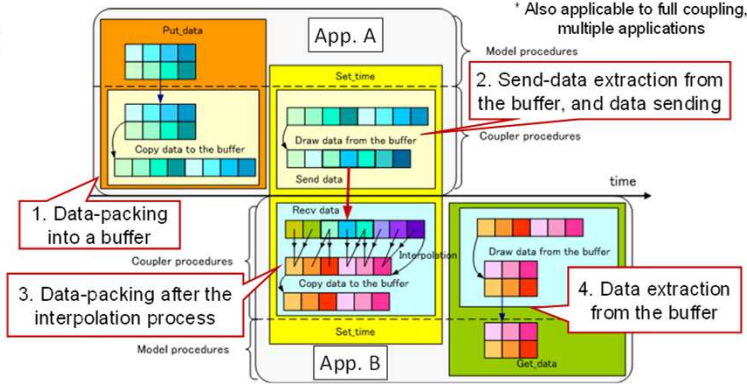
全地球大気・海洋シミュレーション（気候・気象）

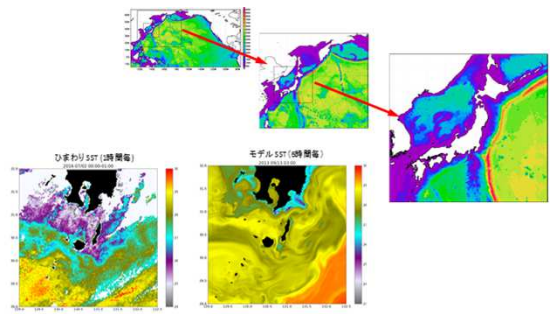
東大大気海洋研究所，東大理学系研究科等







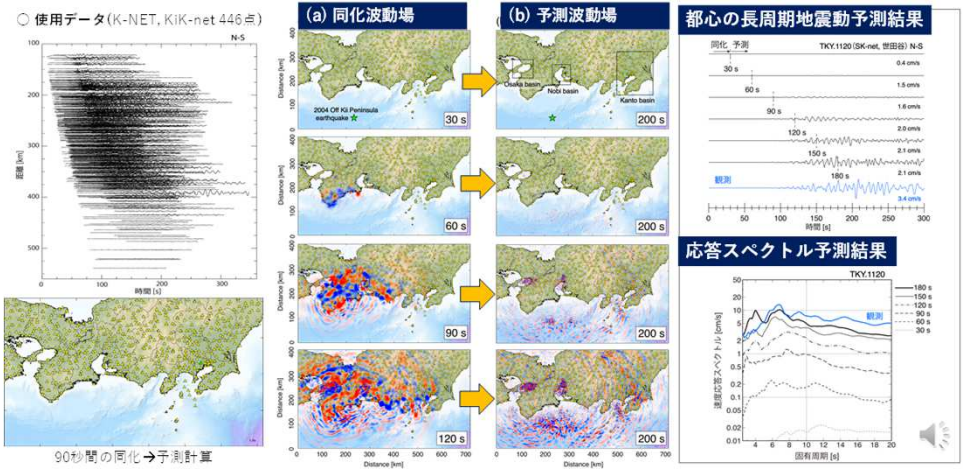


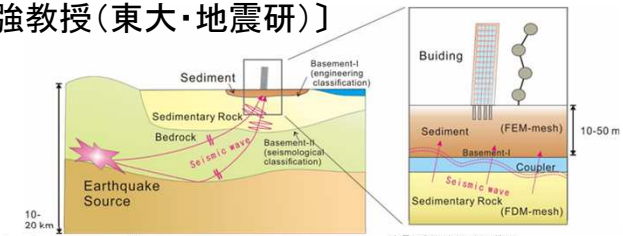


〔画像提供：佐藤正樹教授・
羽角博康教授（東大・大気海洋研）〕

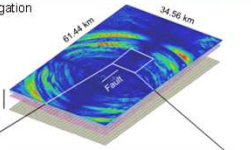
地震シミュレーション・地殻変動
東大地震研究所，東大理学系研究科等

〔画像提供：古村孝志教授・
市村強教授（東大・地震研）〕

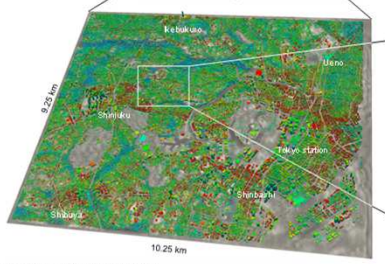


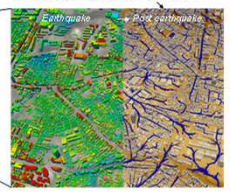


a) Earthquake wave propagation

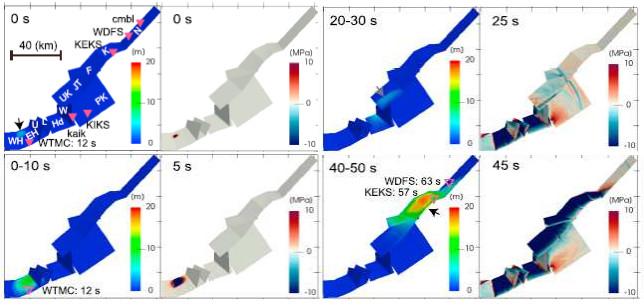


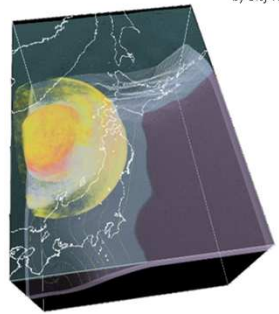
c) Resident evacuation

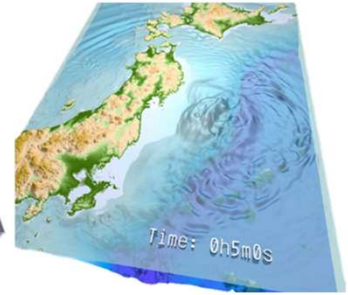


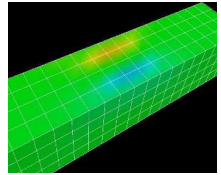


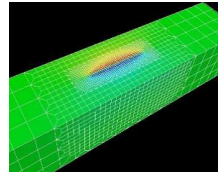
b) City response simulation





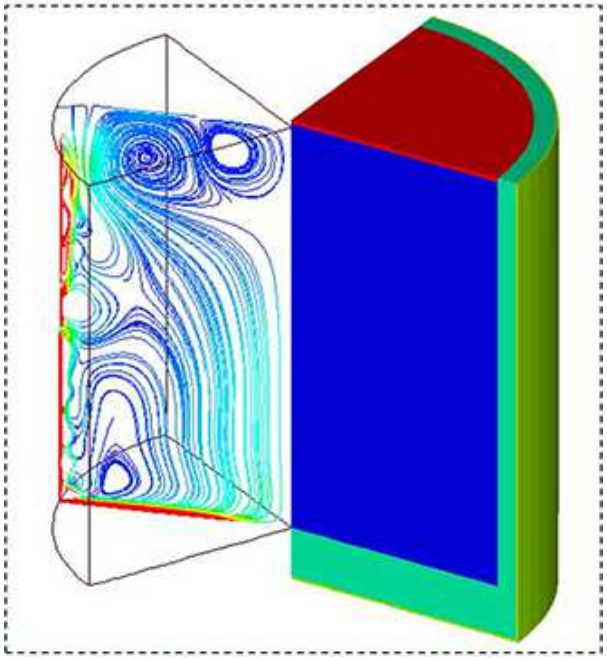


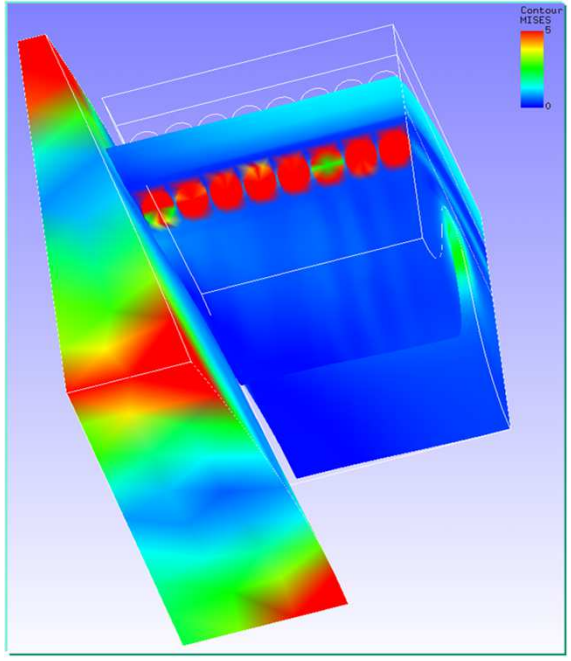




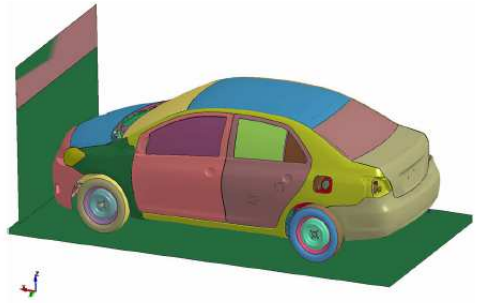
〔画像提供：
安藤亮輔准教授
（東大・理学系）〕

ものづくり分野（流体シミュレーション，構造解析など）
様々な大学・研究機関・企業

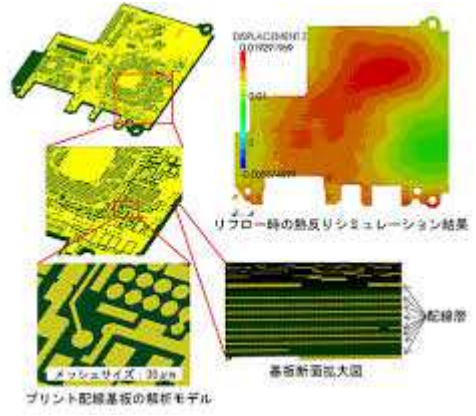




〔画像提供：奥田洋司教授（東京大学新領域創成科学専攻）〕



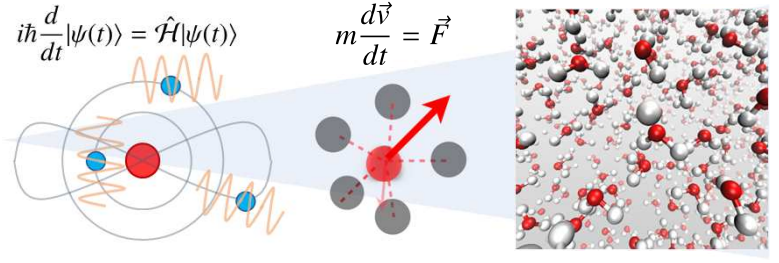
〔画像提供：日本自動車工業会（JAMA）〕



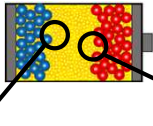
材料科学・物性科学・分子科学

東大工，阪大基礎工，物質・材料研究機構 他 様々な大学・研究機関

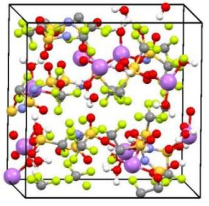
$$i\hbar\frac{d}{dt}|\psi(t)\rangle = \hat{\mathcal{H}}|\psi(t)\rangle \qquad m\frac{d\vec{v}}{dt} = \vec{F} \qquad S = k\ln W$$



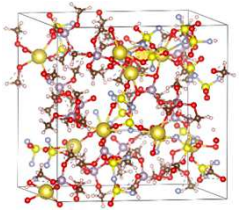
DFT-MDによる蓄電池用新型電解液のミクロ物性解明



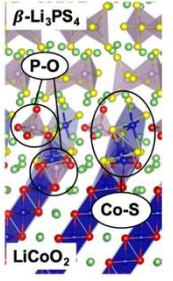
全固体電池の電極-電解質界面のDFTミクロ解析



高濃度水系電解液
Nature Energy 2016



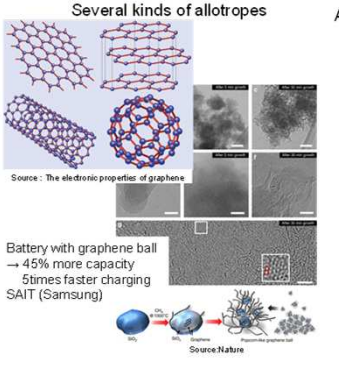
高濃度消火性電解液
Nature Energy 2018



Chem. Mater. 2020

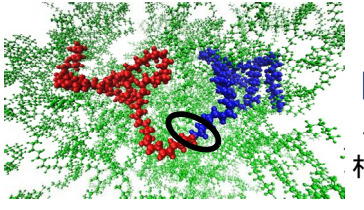
〔画像提供：山田研究室(東大工)，館山グループ（物材機構）〕

ディラック電子系・トポロジカル絶縁体のtight-binding計算



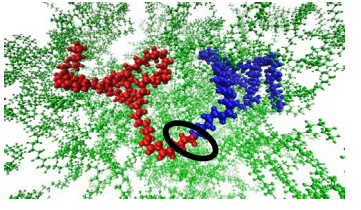
〔画像提供：Gerhard Wellein教授（FAU，Germany）〕

ポリマーの相溶性判定のための全原子自由エネルギー計算



逐次伸張

モノマーごとの相互作用考慮

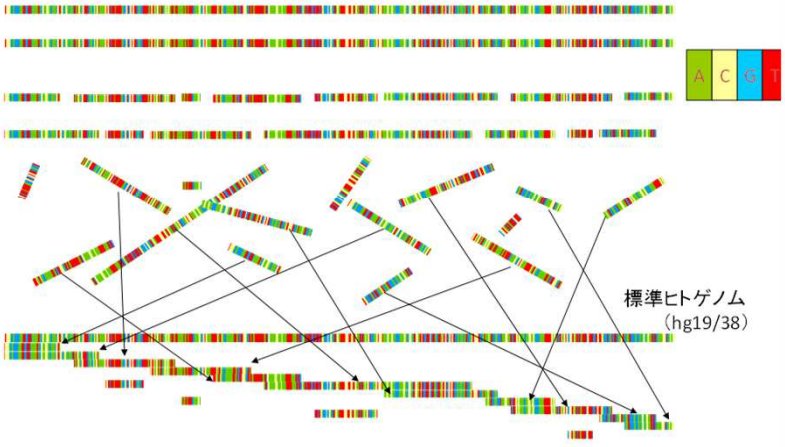


Macromolecules 2020

〔画像提供：松林研究室（阪大）〕

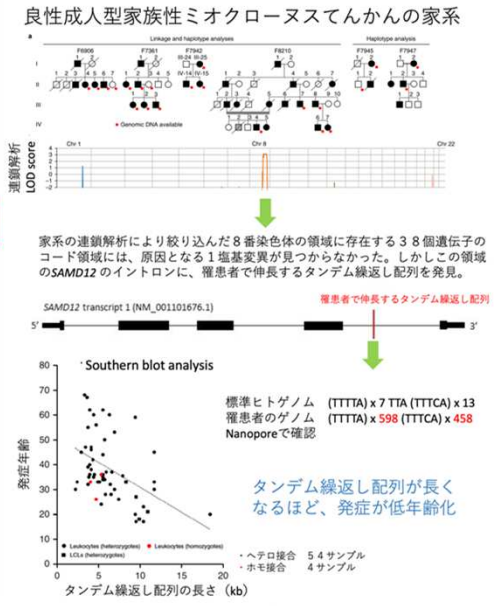
バイオインフォマティックス:ゲノム解析
東大新領域創成科学研究科 等

ヒト個人ゲノムはどのように再解読するか？



長さ 100～150 塩基のDNA 断片を 10～20億本収集(ヒト1人当たり)
接尾辞配列(suffix array), Burrows-Wheeler 変換等が活用される

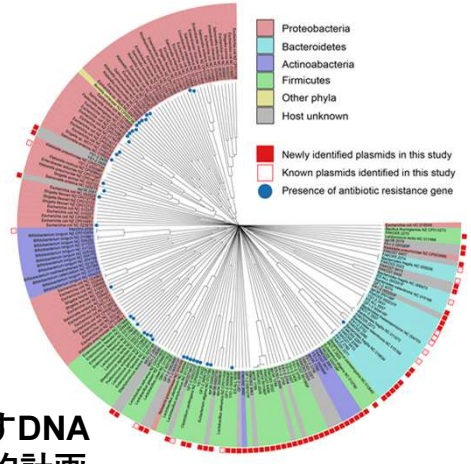
〔画像提供:
森下真一教授(東京大学新領域創成科学研究科)〕



Nature Genetics 2018

新たに発見された疾患を引き起こすDNAの繰返し配列伸長異常：様々な動的計画法，de Bruijn グラフ探索等が活用

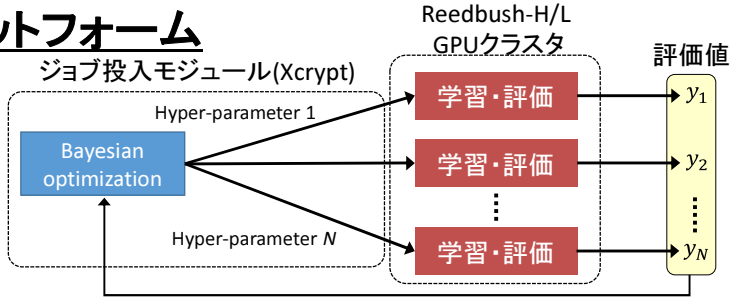
ヒト腸内細菌叢から発見された多様なプラスミド・ファージ配列の全貌



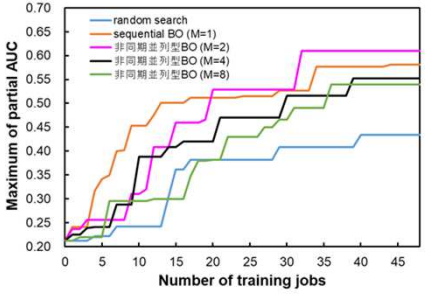
Microbiome 2019

バイオインフォマティックス：医療画像処理
東大病院等

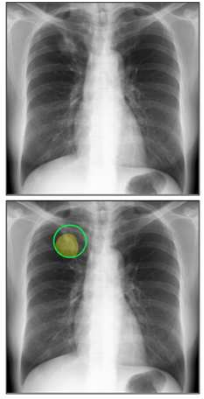
深層学習自動チューニングプラットフォーム



胸部X線写真の肺腫瘤検出



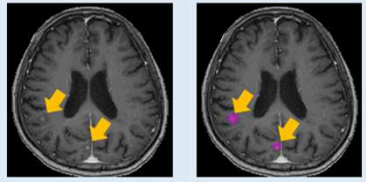
学習ジョブ数と評価値(partial AUC)の最大値との関係



上：元画像、下：検出結果
（黄、緑丸：病変領域）

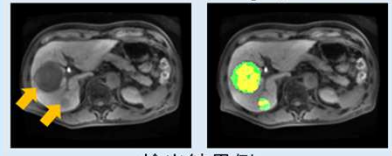
開発中のソフトウェア

頭部造影MR画像の転移性脳腫瘍検出
村田, JAMIT2018



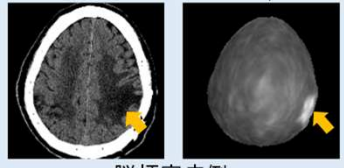
検出結果例
左：元画像、右：検出結果(マゼンダ)

造影MR画像の肝結節性病変検出
Takenaga T, CARS 2018



検出結果例
（肝細胞がん、左：元画像、右：検出結果）
●:検出、●:過検出、●:未検出

頭部救急CT画像の異常検知
Sato D, SPIE MI 2018



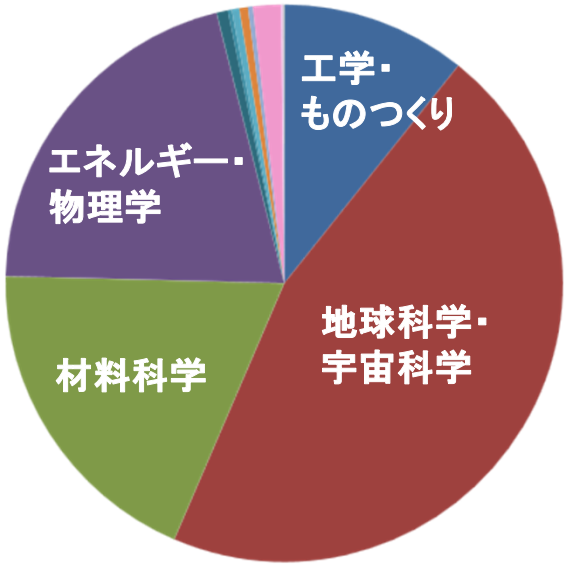
脳梗塞症例
（左：元画像、右：異常度マップ）

胸部X線画像の異常強調
花岡, MAIAMI 2019, JSAIMI 2020

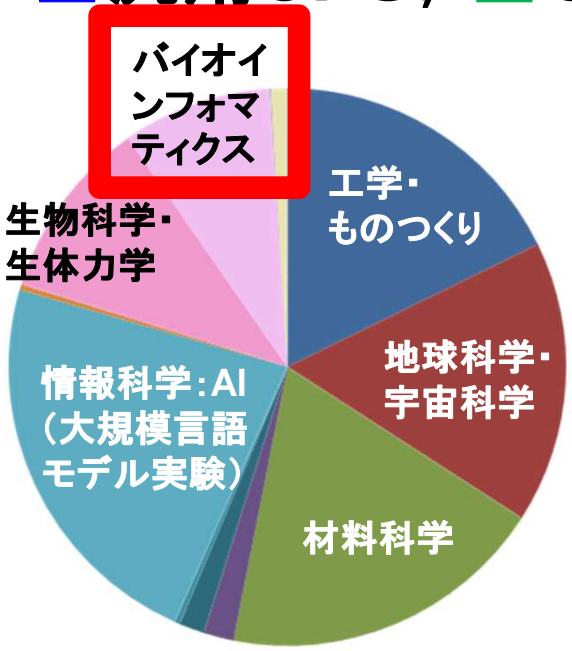
強調画像例
（左：元画像、右：強調結果、矢印：肺腫瘤）

Nomura Y, J Supercomput. 20 Jan 2020 (Epub ahead)

2024年度分野別計算資源利用割合
Wisteria/BDEC-01 ■汎用CPU, ■GPU



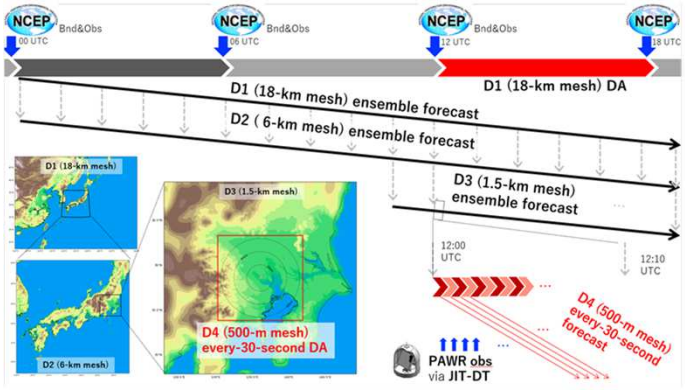
Odyssey
A64FX



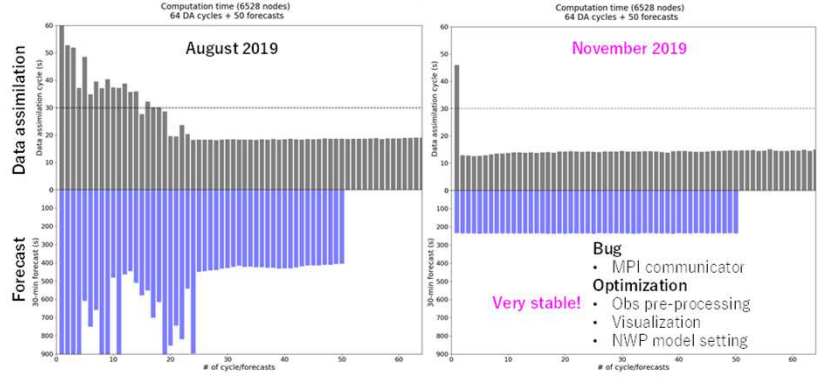
Aquarius
A100

- 工学・ものつくり
- 地球科学・宇宙科学
- 材料科学
- エネルギー・物理学
- 情報科学：システム
- 情報科学：アルゴリズム
- 情報科学：AI
- 教育
- 産業利用
- 生物科学・生体力学
- バイオインフォマティクス
- 社会科学・経済学
- データ科学・データ同化

ゲリラ豪雨予測のリアルタイム実証実験 （理化学研究所）

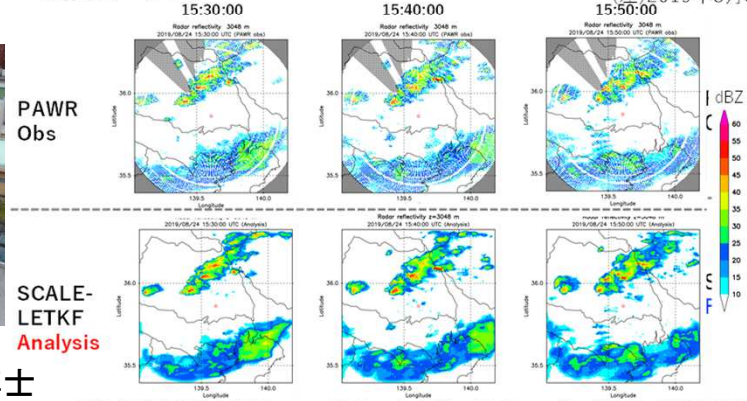


全体のワークフロー

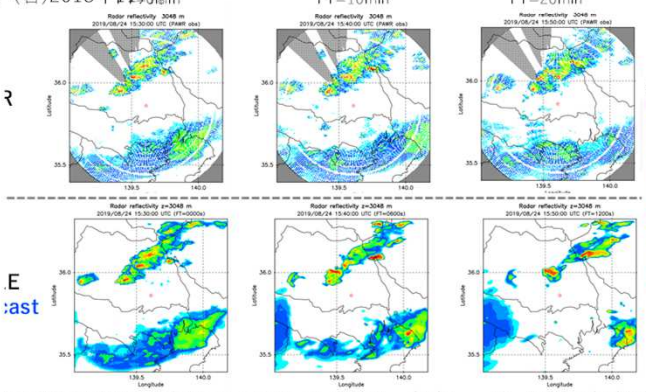


計算性能の向上。上段はデータ同化、下段は30分予報にかかった時間(秒)。
(左)2019年8月、(右)2018年11月。

〔画像提供：三好建正博士
（理化学研究所）〕



2019年8月24日の事例ついてのテスト結果。(上)レーダー観測と(下)SCALE-LETKFによる解析で得られたレーダー反射強度(dBZ)を示す。



2019年8月24日の事例ついてのテスト結果。(上)レーダー観測と(下)SCALE-LETKFによる予報で得られたレーダー反射強度(dBZ)を示す。

- スーパーコンピュータとは
- 第3の科学：計算科学
- **シミュレーションと連立一次方程式**
- 行列とベクトル
- 様々な利用分野
- 数値線形代数とその実用例

様々な自然現象を記述する偏微分方程式

Partial Differential Equation: PDE

- 2階の偏微分方程式で記述される現象は多い：熱，流れ，変形
- 楕円型
  - 定常熱伝導，ポアソン・ラプラス方程式 $$\lambda\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}\right)+Q=0$$
  - ヘルムホルツ方程式 $$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}+k^2 u=0$$
- 放物型
  - 非定常熱伝導方程式 $$\rho c\frac{\partial u}{\partial t}=\lambda\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}\right)+Q$$
- 双曲型
  - 波動方程式 $$\frac{\partial^2 u}{\partial t^2}=c^2\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}\right)$$
  - 非定常移流方程式 $$\frac{\partial u}{\partial t}=c\left(\frac{\partial u}{\partial x}+\frac{\partial u}{\partial y}+\frac{\partial u}{\partial z}\right)+S$$

感染症の数理モデル：SIRモデル：常微分方程式

感受性保持者（Susceptible）・感染者（Infected）・免疫保持者（Recovered）/隔離者（Removed）

$$\frac{dS}{dt}(t) = -\beta S(t) I(t)$$

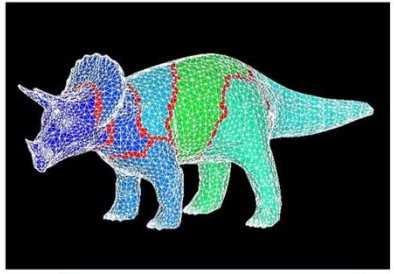
$$\frac{dI}{dt}(t) = \beta S(t) I(t) - \gamma I(t)$$

$$\frac{dR}{dt}(t) = \gamma I(t)$$

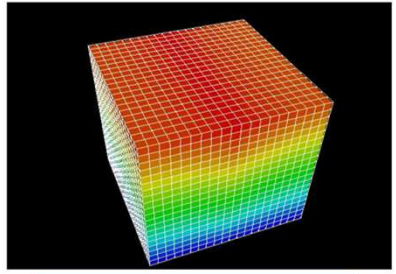
- β　感染率
- γ　回復率（隔離率）
  - $1/\gamma$は平均感染期間

偏微分方程式の解法

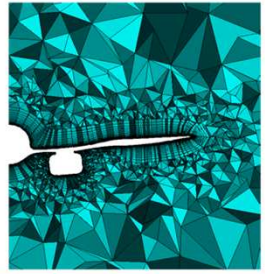
- 非線形方程式，複雑形状等⇒解析解は存在しない
- 空間を，メッシュ，格子，粒子（mesh, grid, particle）に分割し，計算機を使用して数値的に解くのが一般的
  - 細かいメッシュほど計算量は多いが精度の良い解



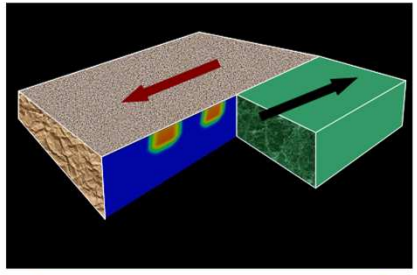
有限要素法
Finite Element Method
FEM



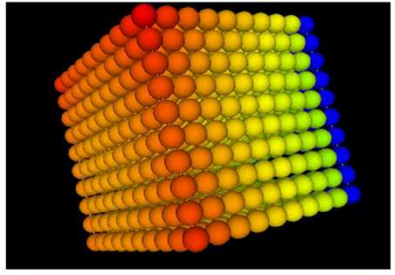
差分法
Finite Difference Method
FDM



有限体積法
Finite Volume Method
FVM



境界要素法
Boundary Element Method
BEM



個別要素法
Discrete Element Method
DEM

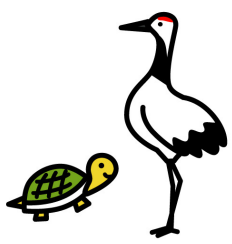
偏微分方程式の数値解法⇒連立一次方程式

- 偏微分方程式の数値解法等による計算科学シミュレーションは，一般に，最終的に大規模連立一次方程式 $Ax=b$ を解くことに帰着される（未知数：数百万～数億，数十億以上）。
  - 重要かつ時間がかかる
  - 計算時間全体の90%以上を占めることもある
    - 高速に安定に求解することが非常に重要

偏微分方程式の数値解法⇒連立一次方程式

- 偏微分方程式の数値解法等による計算科学シミュレーションは，一般に，最終的に大規模連立一次方程式 $Ax=b$ を解くことに帰着される（未知数：数百万～数億，数十億以上）。
  - 重要かつ時間がかかる
  - 計算時間全体の90%以上を占めることもある
    - 高速に安定に求解することが非常に重要
- 鶴亀算：鶴と亀の合計が10匹，足の数の合計が28本
  - 鶴と亀は何匹ずつ？
  - 鶴は2本足：x_1匹，亀は4本足：x_2匹　⇒　二元一次方程式

$$x_1 + x_2 = 10$$
$$2x_1 + 4x_2 = 28$$



$$x_1 = 6, x_2 = 4$$

鶴　　亀

1974年頃小学校で学んだ解法

- 掃き出し法，またはガウスの消去法に相当

$$x_1 + x_2 = 10$$
$$2x_1 + 4x_2 = 28$$

第1式を2倍

$$2x_1 + 2x_2 = 20$$
$$2x_1 + 4x_2 = 28$$

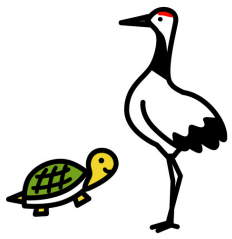
第2式から
第1式を引く

$$2x_2 = 8$$

$$x_2 = 4$$

$$x_1 = 10 - x_2 = 6$$

$$x_1 + x_2 = 10$$
$$2x_1 + 4x_2 = 28$$



$$x_1 = 6, x_2 = 4$$

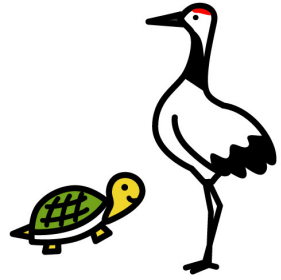
鶴　　亀

- スーパーコンピュータとは
- 第3の科学：計算科学
- シミュレーションと連立一次方程式
- **行列とベクトル**
- 数値線形代数とその実用例

スーパーコンピュータは何をするのか？

- 大量の計算
  - Odyssey　　26 PFLOPS：　2京5,952兆回（世界73位）
  - Miyabi-G　　73 PFLOPS：　7京2,800兆回（世界37位）
  - 富岳　　537 PFLOPS：　53京7,000兆回（世界7位，アジア1位）
  - El Captain 2,746 PFLOPS：274京6,000兆回（世界1位）（=2.746 EFLOPS）

- 連立一次方程式求解（巨大な鶴亀算），固有値計算など，「行列・ベクトル」に関わる演算が主流
- 探索等も行列・ベクトル演算（一次変換）が関わっている
- 機械学習，AIの計算も行列・ベクトル演算
- 量子コンピュータの計算も実は行列・ベクトル演算



行列とベクトル（1/3）

- 行列（Matrix），ベクトル（Vector）
- 「行列・ベクトル」は筆者が高校生だった45年以上前は高2の数学（数Ⅱ）で勉強，その魅力に取り憑かれた
- 下記のような未知数n個のn元一次方程式を考える（x_i：未知数，a_{ij}：係数（既知），b_j：右辺（既知），$1 \leqq i,j \leqq n$）：

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\
a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\
&\vdots \\
a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n
\end{aligned}
$$

- 「行列」とは，数や記号や式などを「行（Row）」と「列（Column）」に沿って矩形状に配列したものである

行列とベクトル(2/3):連立一次方程式

- ベクトルは，n個の変数の組で，上記の連立一次方程式の未知数x_i，右辺b_jをまとめて，未知数ベクトル，右辺ベクトルとして扱うことができる。
- ここで下記のように係数行列(Coefficient Matrix):$\boldsymbol{A}$，右辺ベクトル:$\boldsymbol{b}$，未知数ベクトル:$\boldsymbol{x}$を定義すると，前頁の連立一次方程式は行列とベクトルの積によって$\boldsymbol{Ax=b}$と表すことができる。

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

$$\boldsymbol{A} \qquad \boldsymbol{x} \qquad \boldsymbol{b}$$

行列とベクトルの演算

- 行列ベクトル積

第2行

$$\begin{bmatrix} a & a & a & a \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a & a & a & a \\ a & a & a & a \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} y \\ y_2 \\ y \\ y \end{bmatrix}$$

$$a_{21} \times x_1 + a_{22} \times x_2 + a_{23} \times x_3 + a_{24} \times x_4 = y_2$$

- 行列・行列積

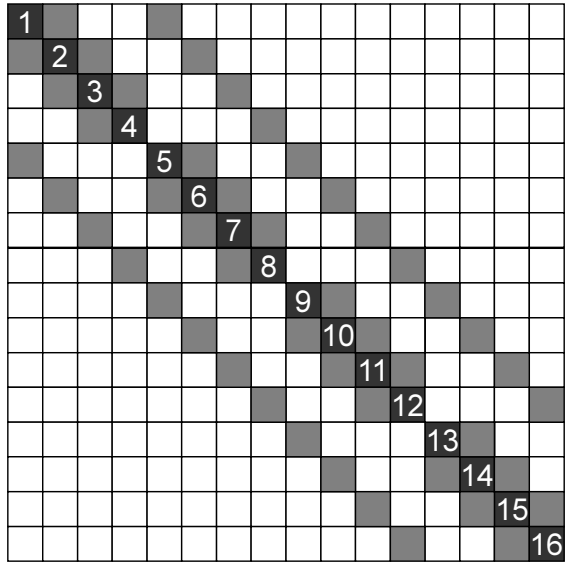
第3列

第2行

$$\begin{bmatrix} a & a & a & a \\ a & a & a & a \\ a & a & a & a \\ a & a & a & a \end{bmatrix} \begin{bmatrix} b & b & b & b \\ b & b & b & b \\ b & b & b & b \\ b & b & b & b \end{bmatrix} = \begin{bmatrix} c & c & c & c \\ c & c & c_{23} & c \\ c & c & c & c \\ c & c & c & c \end{bmatrix}$$

行列とベクトル(3/3):単位行列・逆行列

■対角成分
■非ゼロ非対角成分
□ゼロ成分



- 対角成分(a_{ii})が全て1，他の成分が全て0となるような行列を「単位行列(Identity Matrix)」と呼ぶ
  - 任意のベクトルに単位行列を乗じてもベクトルは不変である。
  - 単位行列はIと書かれることが多い

- $AX=I$が成立するとき行列XはAの逆行列(Inverse Matrix)A^{-1}であると言う
  - 逆行列は一般の数値演算における逆数に相当

$$I = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

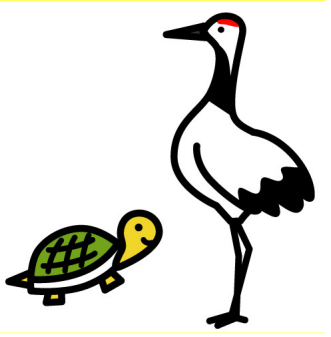
鶴亀算を「逆行列」を使って解く

- 鶴と亀の合計が10匹，足の数の合計が28本，鶴と亀は何匹ずつ？
- 鶴は2本足：x_1匹，亀は4本足：x_2匹

$$\begin{aligned} x_1 + x_2 &= 10 \\ 2x_1 + 4x_2 &= 28 \end{aligned}$$

$$\boldsymbol{Ax} = \boldsymbol{b}, \quad \boldsymbol{A} = \begin{bmatrix} 1 & 1 \\ 2 & 4 \end{bmatrix}$$

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} 10 \\ 28 \end{bmatrix}$$



- 逆行列の公式

$$A^{-1} = \frac{1}{\det(A)} \begin{bmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{bmatrix} = \frac{1}{a_{11} \times a_{22} - a_{12} \times a_{21}} \begin{bmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{bmatrix}$$

$$\boldsymbol{Ax} = \boldsymbol{b} \Rightarrow \boldsymbol{A}^{-1}\boldsymbol{Ax} = \boldsymbol{A}^{-1}\boldsymbol{b} \Rightarrow \left(\boldsymbol{A}^{-1}\boldsymbol{A}\right)\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b} \Rightarrow \boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$$

$$\begin{bmatrix} 1 & 1 \\ 2 & 4 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 10 \\ 28 \end{bmatrix} \qquad \boldsymbol{A}^{-1} = \frac{1}{1\times 4 - 1\times 2}\begin{bmatrix} 4 & -1 \\ -2 & 1 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 4 & -1 \\ -2 & 1 \end{bmatrix}$$

$$\boldsymbol{A}^{-1}\boldsymbol{A} = \frac{1}{2}\begin{bmatrix} 4 & -1 \\ -2 & 1 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 2 & 4 \end{bmatrix}$$

$$= \frac{1}{2}\begin{bmatrix} 4\times 1 - 1\times 2 & 4\times 1 - 1\times 4 \\ -2\times 1 + 2\times 2 & -2\times 1 + 1\times 4 \end{bmatrix}$$

$$= \frac{1}{2}\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \boldsymbol{I}$$

単位行列

$$\boldsymbol{A}^{-1}\boldsymbol{b} = \boldsymbol{x} = \frac{1}{2}\begin{bmatrix} 4 & -1 \\ -2 & 1 \end{bmatrix}\begin{bmatrix} 10 \\ 28 \end{bmatrix}$$

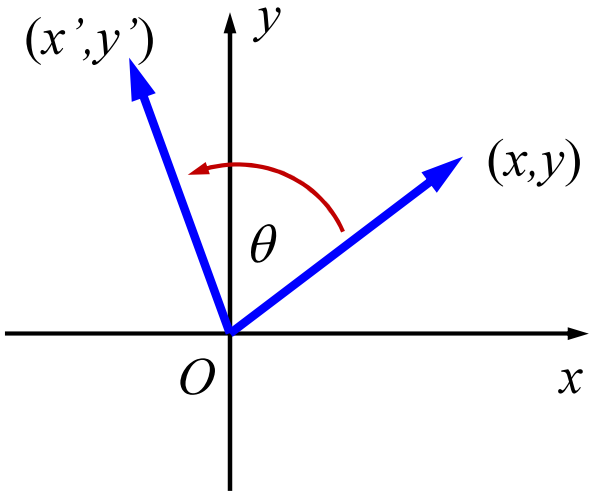
$$= \frac{1}{2}\begin{bmatrix} 4\times 10 - 1\times 28 \\ -2\times 10 + 1\times 28 \end{bmatrix}$$

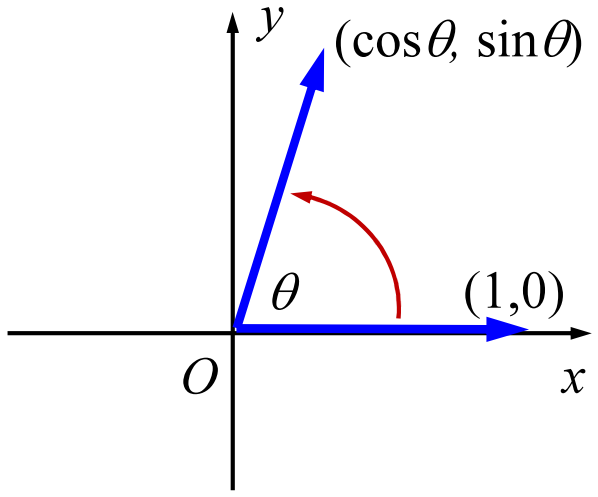
$$= \frac{1}{2}\begin{bmatrix} 12 \\ 8 \end{bmatrix} = \begin{bmatrix} 6 \\ 4 \end{bmatrix} \begin{matrix} \text{鶴} \\ \text{亀} \end{matrix}$$

計算量多い（N個の未知数に対して$O(N^3)$の演算），Nが増えると大変

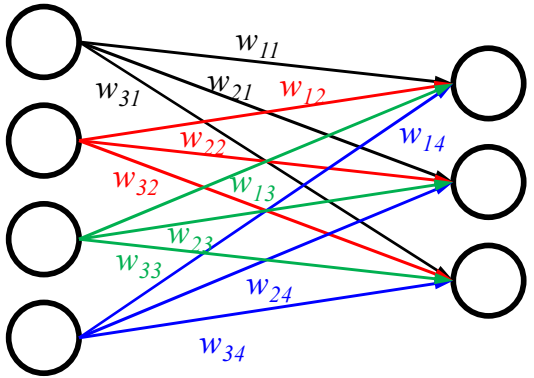
行列の応用：ベクトルの回転（二次元）
行列をかける処理：一次変換（線形変換）

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}\begin{pmatrix} x \\ y \end{pmatrix}=\begin{pmatrix} x\cos\theta - y\sin\theta \\ x\sin\theta + y\cos\theta \end{pmatrix} \qquad \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix}=\begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix}$$





一次変換（線形変換）とニューラルネットワーク



入力ベクトル　重み　出力ベクトル

$\boldsymbol{x}$　$\boldsymbol{W}$　$\boldsymbol{y}$

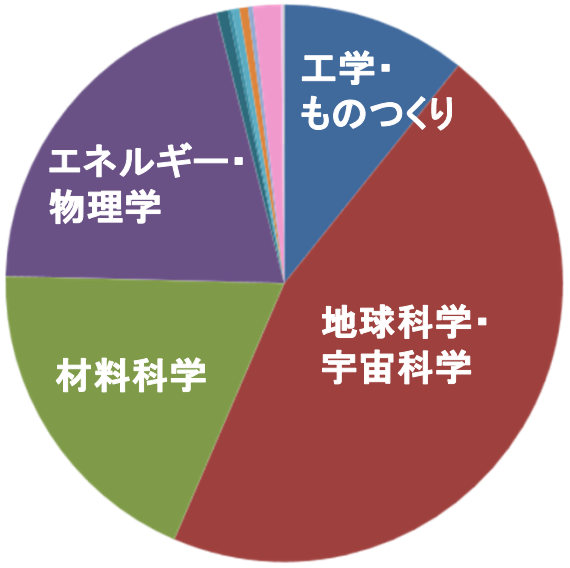
$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}, \quad \boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

$$\boldsymbol{W} = \begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{24} \\ w_{31} & w_{32} & w_{33} & w_{34} \end{bmatrix}$$

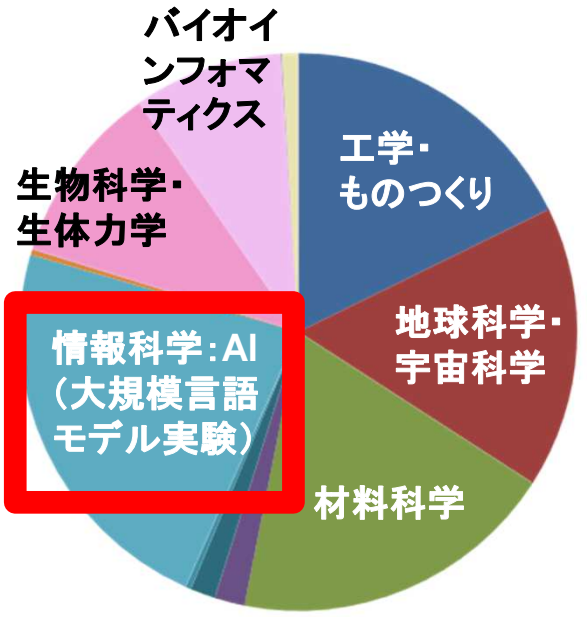
$$\boldsymbol{W}\boldsymbol{x} = \boldsymbol{y}$$

- 人工知能で使用される「ニューラルネットワーク」の計算の基本は，パラメータと重みのかけ算の足し合わせ⇒一次変換（線形変換）

2024年度分野別計算資源利用割合
Wisteria/BDEC-01 ■汎用CPU, ■GPU



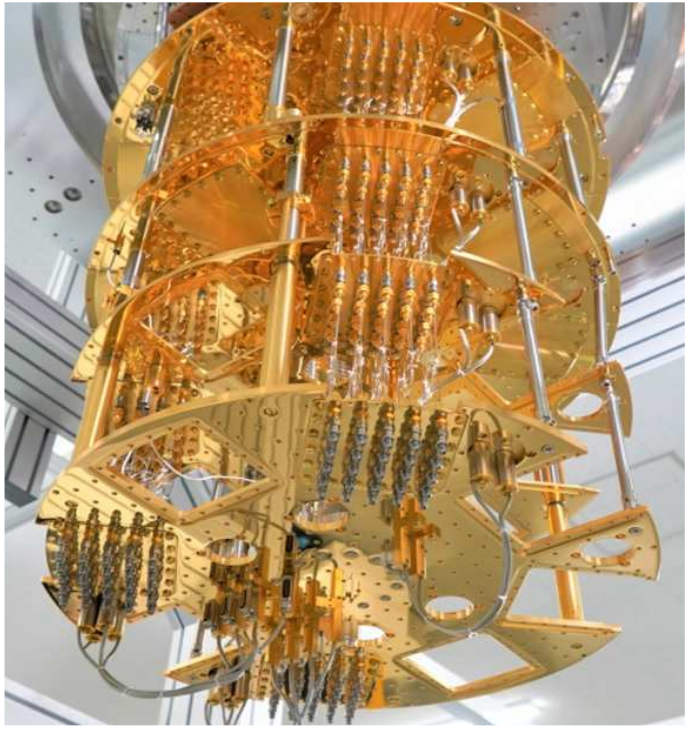
Odyssey
A64FX



Aquarius
A100

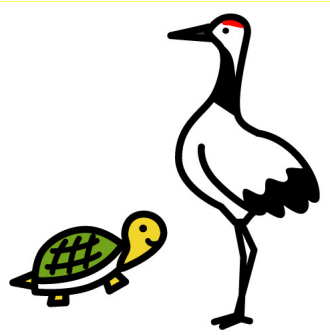
- 工学・ものつくり
- 地球科学・宇宙科学
- 材料科学
- エネルギー・物理学
- 情報科学：システム
- 情報科学：アルゴリズム
- 情報科学：AI
- 教育
- 産業利用
- 生物科学・生体力学
- バイオインフォマティクス
- 社会科学・経済学
- データ科学・データ同化

量子コンピュータでやっている計算も実は一次変換（線形変換）



- スーパーコンピュータとは
- 第3の科学：計算科学
- シミュレーションと連立一次方程式
- 行列とベクトル
- **数値線形代数とその実用例**

数値線形代数：Numerical Linear Algebra



- 連立一次方程式を計算機を使って解くことを研究する学問
  - 巨大な「鶴亀算」の解法に関する研究
  - 様々な分野のシミュレーションに役に立つ（と思う）

- 直接法（Direct Method）
  - ガウスの消去法，「逆行列」を求める方法⇒大規模問題では時間がかかる，メモリ消費量も大
- 反復法（Iterative Method）
  - 繰り返しにより，正解に近づける
  - 大規模問題，並列計算機向け

反復法とは・・・

Linear Equations
連立一次方程式

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

$$\boldsymbol{A} \qquad \boldsymbol{x} \qquad \boldsymbol{b}$$

Initial Solution
初期解

$$\mathbf{x}^{(0)} = \begin{pmatrix} x_1^{(0)} \\ x_2^{(0)} \\ \vdots \\ x_n^{(0)} \end{pmatrix}$$

適当な初期解 $\mathrm{x}^{(0)}$から始めて，繰り返し計算によって真の解に収束(converge)させていく

$$\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots$$

共役勾配法（Conjugate Gradient, CG）：反復法

未知数がN個の場合N回以内に収束することが理論的に証明されている。
前処理（Preconditioning）によって行列の性質を「良く」して速く収束させることも可能。

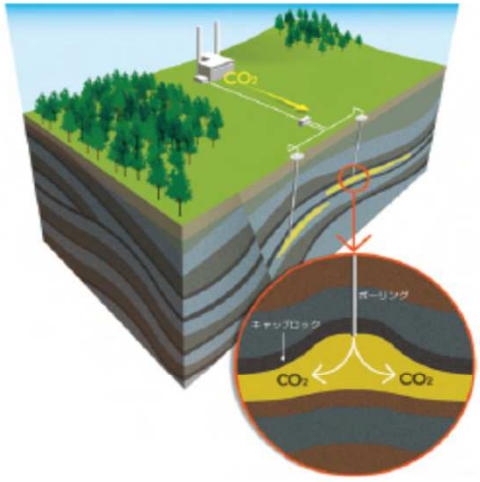
- 行列ベクトル積
- ベクトル内積
- ベクトル定数倍の加減

$x^{(i)}$ ：ベクトル
α_i ：スカラー

```
Compute r(0)= b-[A]x(0)
for i= 1, 2, …
    z(i-1)= r(i-1)
    ρi-1= r(i-1) z(i-1)
    if i=1
      p(1)= z(0)
     else
      βi-1= ρi-1/ρi-2
      p(i)= z(i-1) + βi-1 p(i-1)
    endif
    q(i)= [A]p(i)
    αi  = ρi-1/p(i)q(i)
    x(i)= x(i-1) + αip(i)
    r(i)= r(i-1) - αiq(i)
    check convergence |r|
end
```

二酸化炭素地下貯留シミュレーション
三次元多相流れ＋物質移動

〔画像提供：
山本肇博士（大成建設）〕



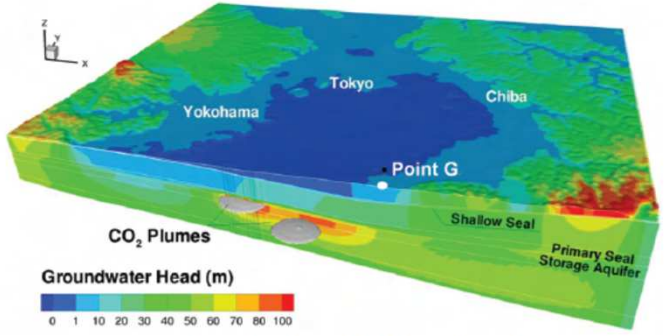
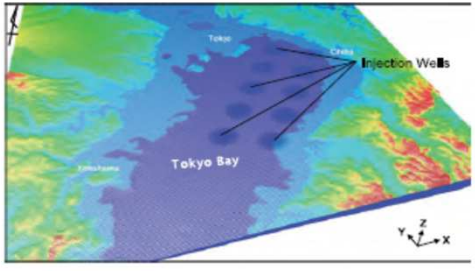
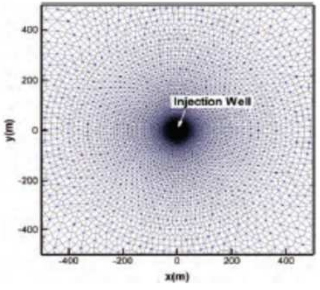
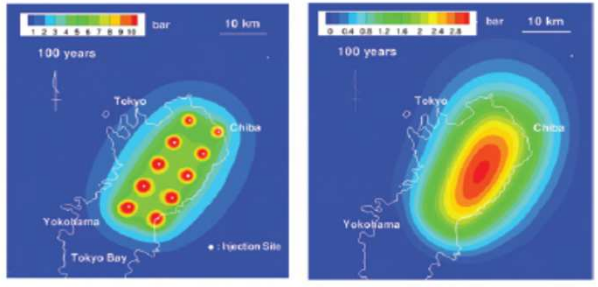


図-4　CO_2 圧入後の地下水圧（全水頭換算）の分布（100 年後）

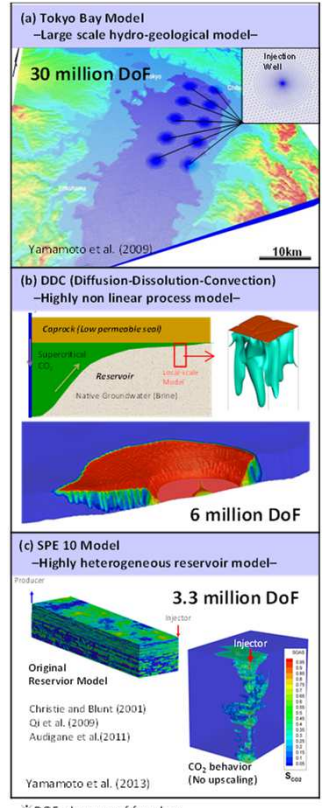






(a) 深部遮蔽層下面　　(b) 浅部遮蔽層下面

図-5　圧力上昇量の平面分布（初期状態からの増分、圧入開始から 100 年後）



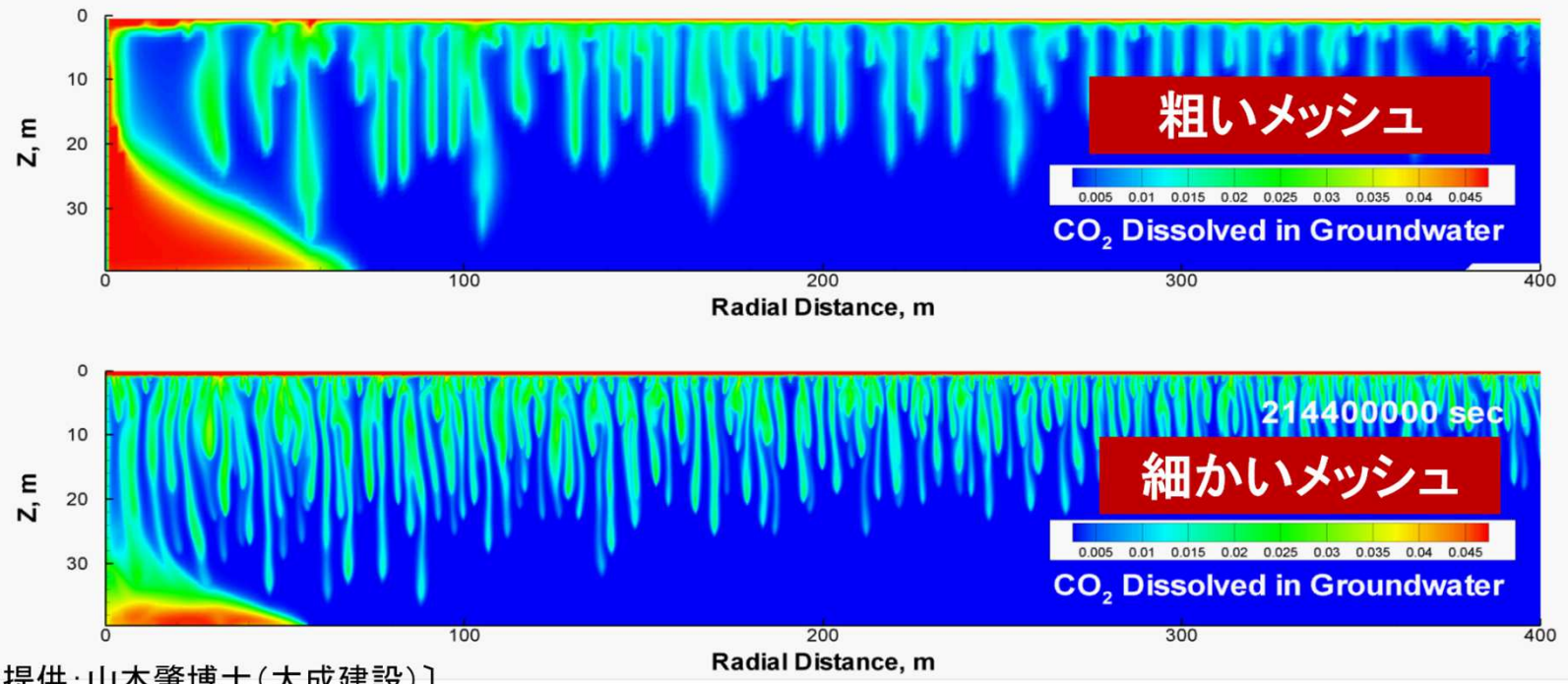
※DOF: degrees of freedom

CO_2が地下水に溶けていく様子

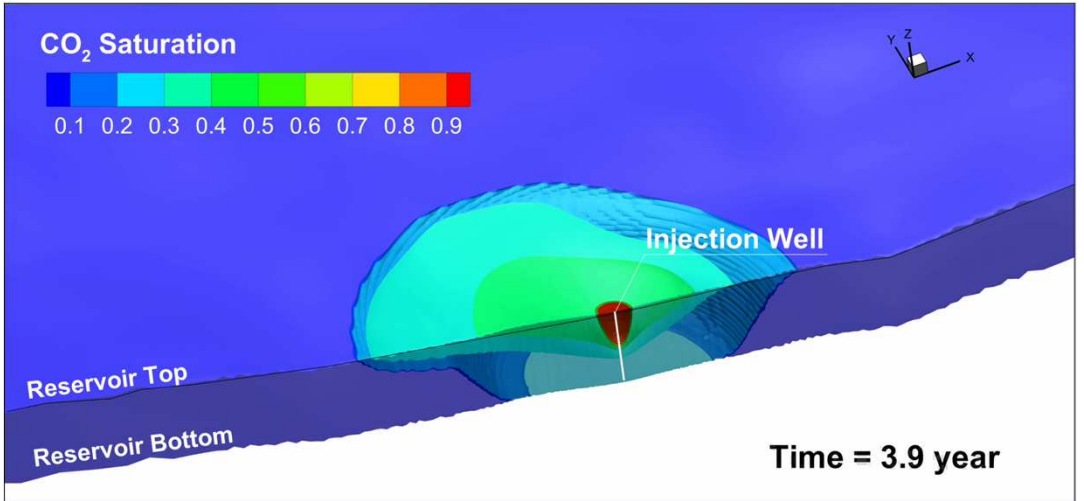
正確な予測のためには細かいメッシュが必要⇒
大規模な計算モデル，連立一次方程式

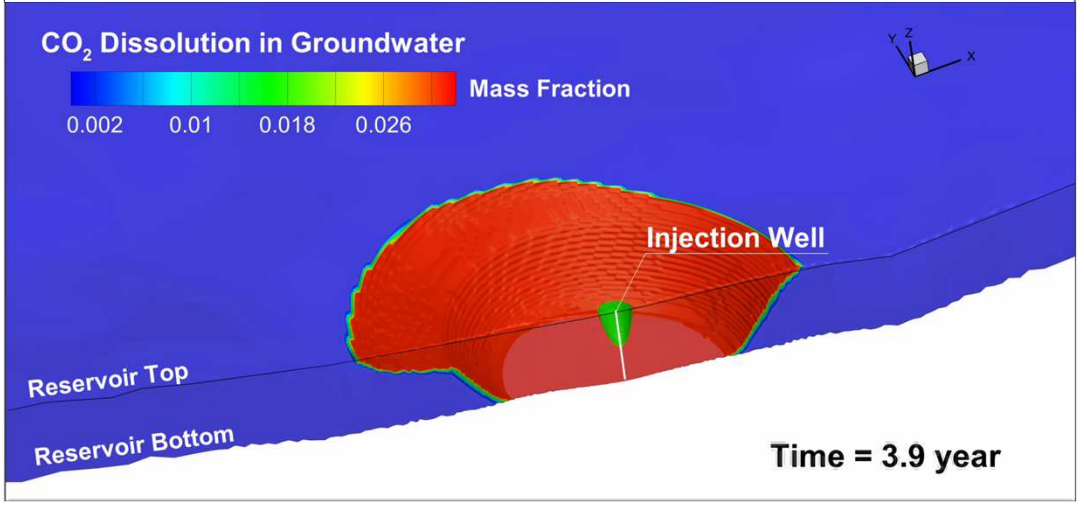
Movie

三次元多相流れ（気液）＋物質移動，という非常に複雑で難しい問題



〔画像提供：山本肇博士（大成建設）〕



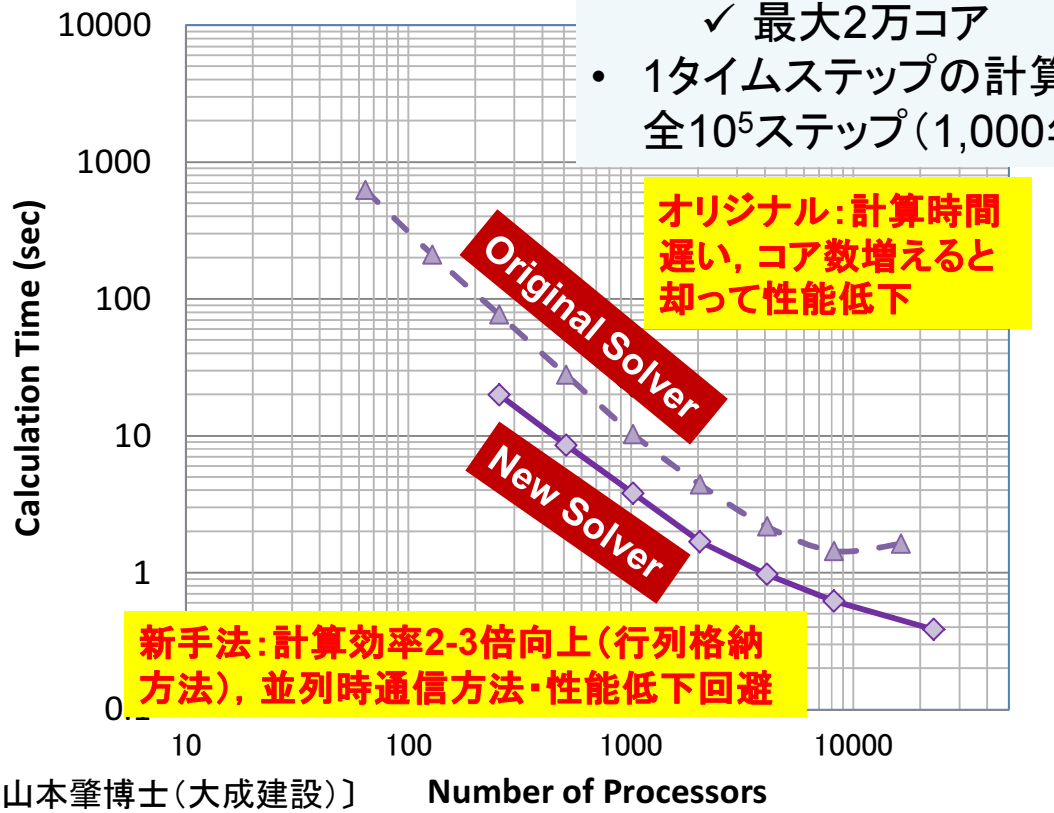


1,000年分の計算時間履歴
未知数：3千万
10^5タイムステップ
（1ステップ3日）

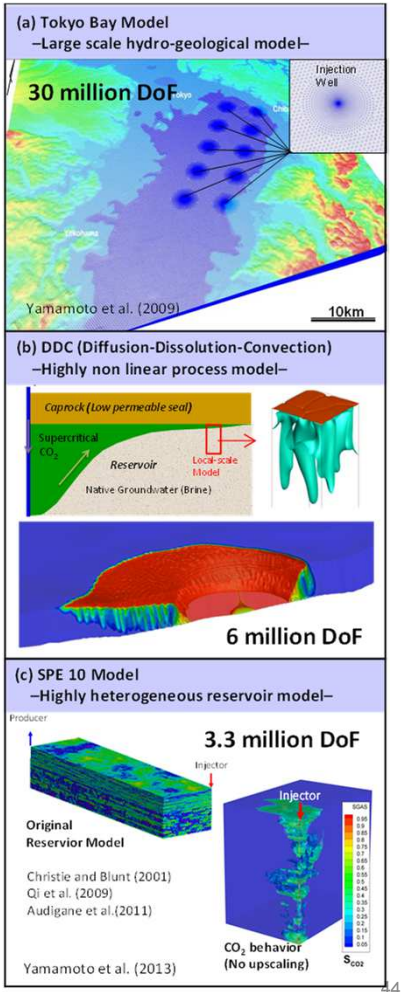
〔画像提供：山本肇博士（大成建設）〕

大規模計算結果
連立一次方程式解法の影響大

- 約3千万自由度
- 前処理付BiCGSTAB法
- Fujitsu FX10（Oakleaf-FX）
  - ✓ 最大2万コア
- 1タイムステップの計算時間，全10^5ステップ（1,000年）



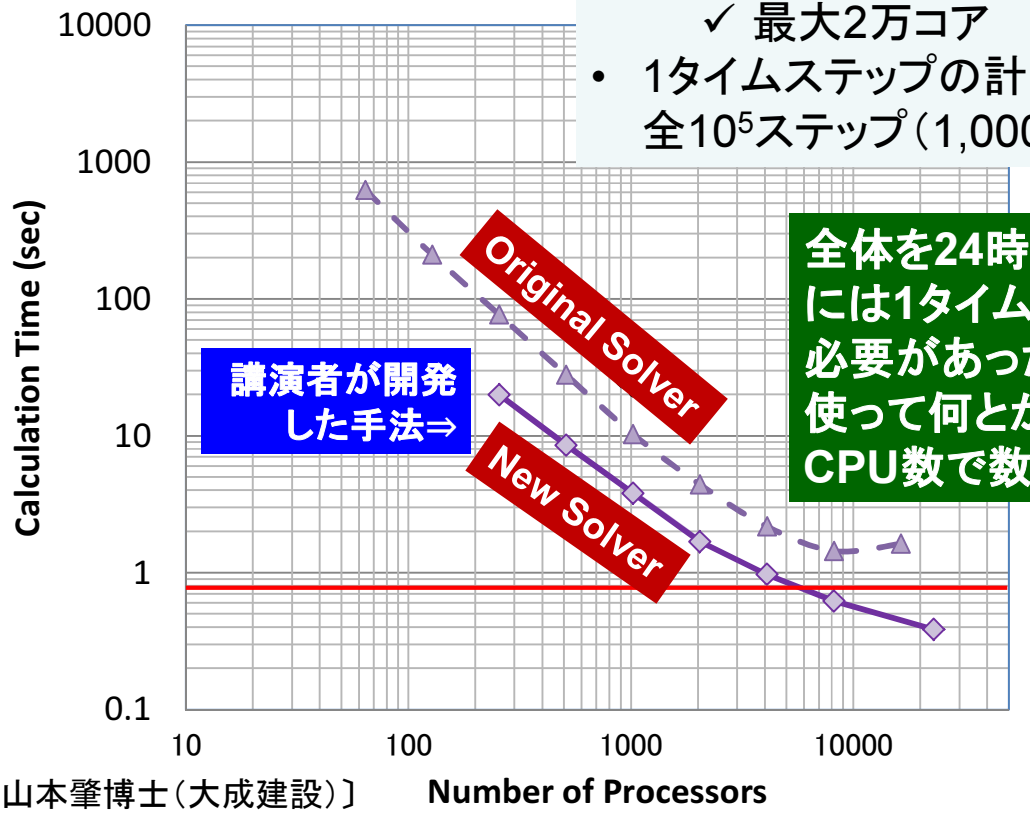
〔画像提供：山本肇博士（大成建設）〕



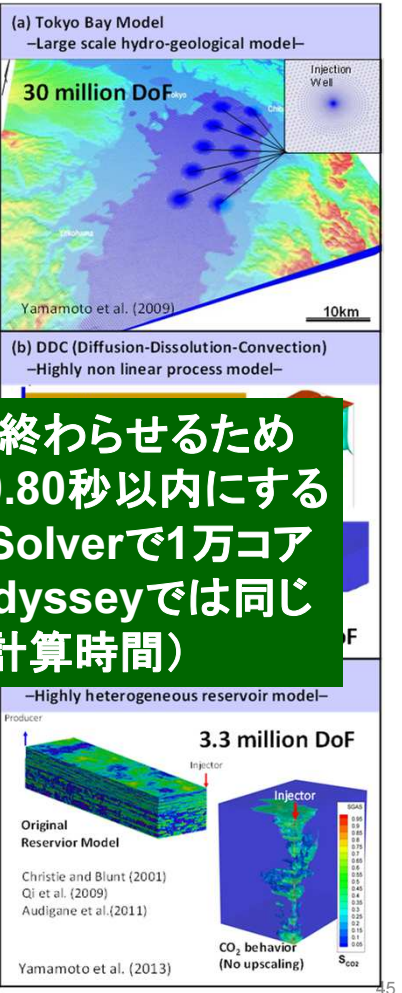
※DOF: degrees of freedom

大規模計算結果
連立一次方程式解法の影響大

- 約3千万自由度
- 前処理付BiCGSTAB法
- Fujitsu FX10（Oakleaf-FX）
  - ✓ 最大2万コア
- 1タイムステップの計算時間，全10^5ステップ（1,000年）



全体を24時間以内で終わらせるためには1タイムステップ0.80秒以内にする必要があった。New Solverで1万コア使って何とか達成（Odysseyでは同じCPU数で数分の1の計算時間）



※DOF: degrees of freedom

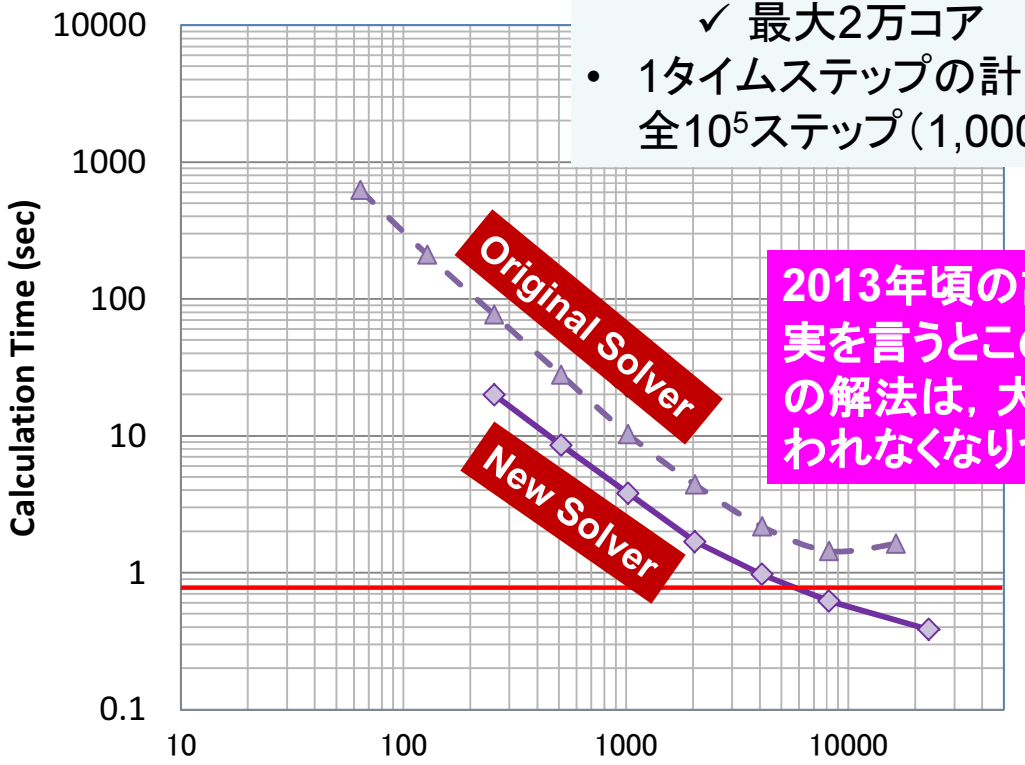
〔画像提供：山本肇博士（大成建設）〕

まとめ

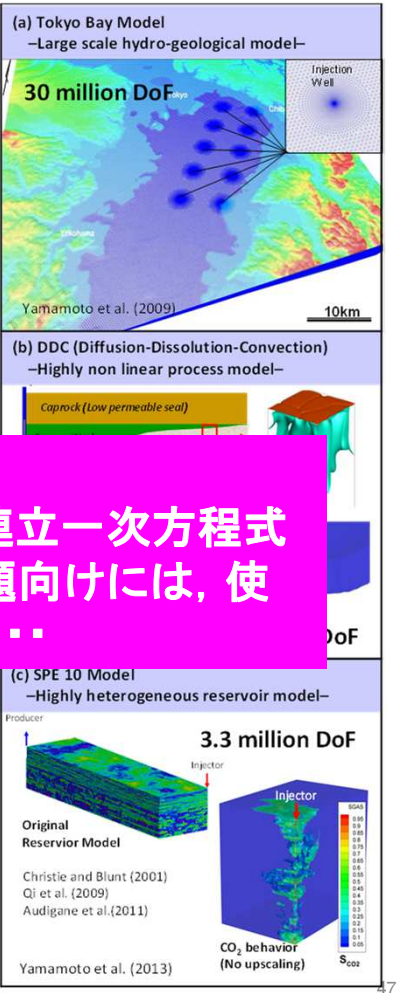
- 第3の科学：計算科学，シミュレーション
  - 偏微分方程式の数値解法等による計算科学シミュレーションは，一般に，最終的に大規模連立一次方程式 Ax=b を解くことに帰着される。
  - 重要かつ時間がかかる
  - 計算時間全体の90%以上を占めることもある
- 数値線形代数（Numerical Linear Algebra）
  - 連立一次方程式を計算機を使って解くことを研究する学問
- **高速な連立一次方程式解法によって，計算科学による新たな科学的発見の促進，加速が期待される**
  - **CO_2地下貯留シミュレーションの事例**
  - **数学，応用数学の計算科学への貢献**
- **「行列・ベクトル」を勉強しよう！**

大規模計算結果
連立一次方程式解法の影響大

- 約3千万自由度
- 前処理付BiCGSTAB法
- Fujitsu FX10（Oakleaf-FX）
  - ✓ 最大2万コア
- 1タイムステップの計算時間，全10^5ステップ（1,000年）



2013年頃の計算
実を言うとこのような連立一次方程式の解法は，大規模問題向けには，使われなくなりつつある・・・



※DOF: degrees of freedom

〔画像提供：山本肇博士（大成建設）〕

共役勾配法（Conjugate Gradient，CG）：反復法

未知数がN個の場合N回以内に収束することが理論的に証明されている。
前処理（Preconditioning）によって行列の性質を「良く」して速く収束させることも可能。

- 行列ベクトル積
- ベクトル内積
- ベクトル定数倍の加減

$x^{(i)}$ ：ベクトル
α_i ：スカラー

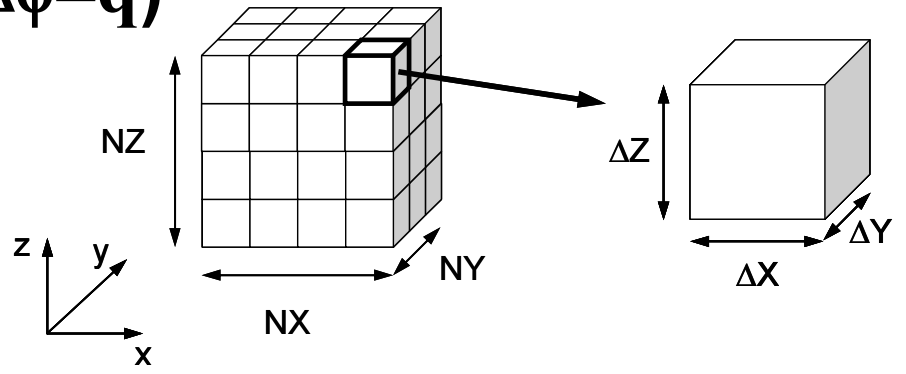
```
Compute r(0)= b-[A]x(0)
for i= 1, 2, …
    z(i-1)= r(i-1)
    ρi-1= r(i-1) z(i-1)
    if i=1
      p(1)= z(0)
     else
      βi-1= ρi-1/ρi-2
      p(i)= z(i-1) + βi-1 p(i-1)
    endif
    q(i)= [A]p(i)
    αi  = ρi-1/p(i)q(i)
    x(i)= x(i-1) + αip(i)
    r(i)= r(i-1) - αiq(i)
    check convergence |r|
end
```

前処理付き共役勾配法（Preconditioned Conjugate Gradient, PCG）
前処理行列[M]を適用して，問題の性質を「改善」

Compute $r^{(0)}= b-[A]x^{(0)}$
for i= 1, 2, …
  $[M]z^{(i-1)}= r^{(i-1)}$
  $\rho_{i-1}= r^{(i-1)}\ z^{(i-1)}$
  if i=1
    $p^{(1)}= z^{(0)}$
  else
    $\beta_{i-1}= \rho_{i-1}/\rho_{i-2}$
    $p^{(i)}= z^{(i-1)} + \beta_{i-1}\ p^{(i-1)}$
  endif
  $q^{(i)}= [A]p^{(i)}$
  $\alpha_i = \rho_{i-1}/p^{(i)}q^{(i)}$
  $x^{(i)}= x^{(i-1)} + \alpha_i p^{(i)}$
  $r^{(i)}= r^{(i-1)} - \alpha_i q^{(i)}$
  check convergence |r|
end

ICCG法（前処理付き共役勾配法の一種）の反復回数
三次元ポアソン方程式($\Delta\phi=q$)

N	反復回数
8^3= 512	19
16^3= 4,096	38
32^3= 32,768	75
64^3= 262,144	146
128^3=2,097,152	290

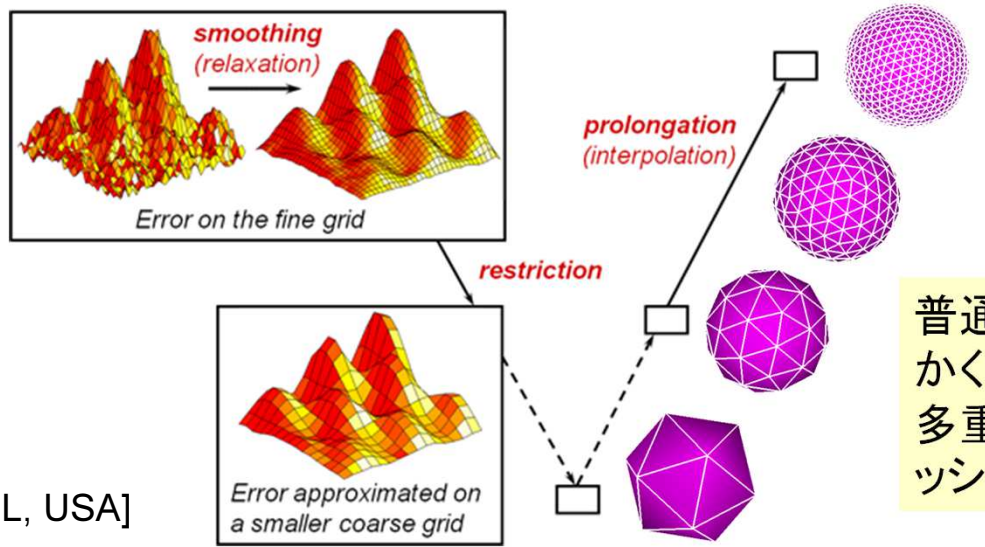


- ICCG = 不完全コレスキー分解前処理＋共役勾配法（CG）
  - 「不完全な直接法」を前処理：広く使用

- 反復回数：3Dでは，概ね$N^{1/3}$に比例（8倍の問題規模で2倍）（問題に依存）
- 問題規模が1000倍：計算時間$1000^{4/3}=10^4$倍
  - ✓ CPUを1,000倍に増やしても計算時間は10倍かかる

多重格子法(Multigrid):夢のアルゴリズム

- 多重格子法(Multigrid)は，計算対象を様々なメッシュ幅に分割したメッシュ系(格子レベル，最も細かいメッシュ系のレベル=1とする)を予め準備しておき，各メッシュ幅(格子幅)に対応した波長の残差成分を減衰させる方法である



[c/o LLNL, USA]

普通はメッシュ(格子)を細かくした方が良いのだが，多重格子法では「粗い」メッシュを積極的に活用

多重格子法(Multigrid):夢のアルゴリズム

- 多重格子法(Multigrid)は，計算対象を様々なメッシュ幅に分割したメッシュ系(格子レベル，最も細かいメッシュ系のレベル=1とする)を予め準備しておき，各メッシュ幅(格子幅)に対応した波長の残差成分を減衰させる方法である。
- 各波長の残差成分を一様に減衰できるため，問題規模によらず反復回数が一様なアルゴリズムを構成できる。
  - 1,000倍の規模の問題を解くのに1,000倍の数のCPUを用意すれば，同じ時間で計算ができる

反復回数:三次元ポアソン方程式

N	ICCG	Multigrid
8^3= 512	19	4
16^3= 4,096	38	6
32^3= 32,768	75	9
64^3= 262,144	146	13
128^3=2,097,152	290	17

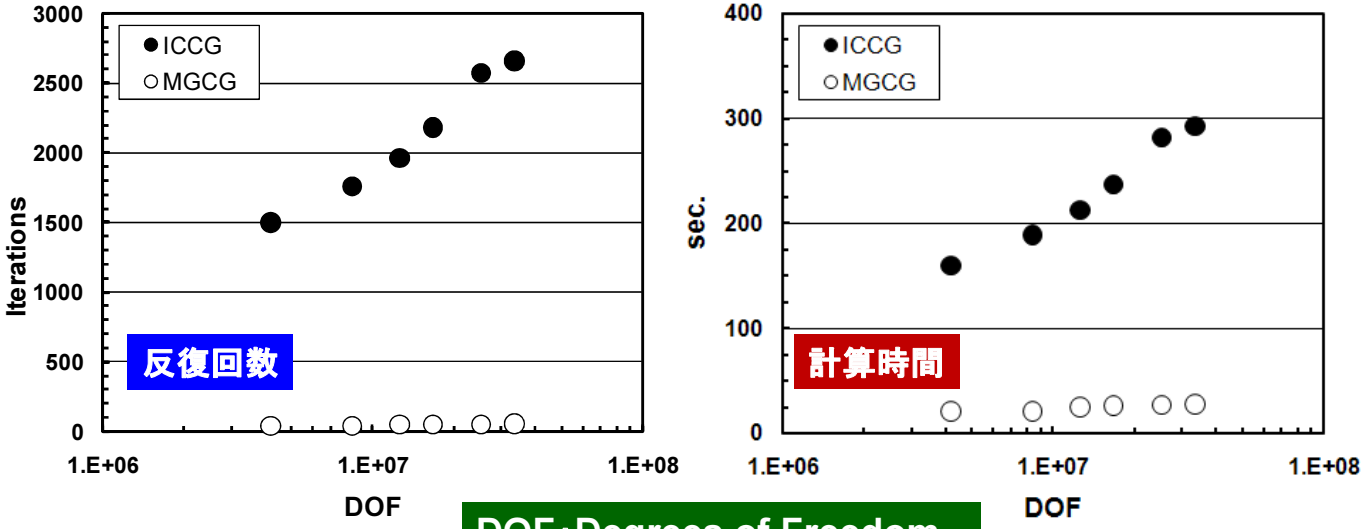
多重格子法(Multigrid):夢のアルゴリズム

- 多重格子法(Multigrid)は,計算対象を様々なメッシュ幅に分割したメッシュ系(格子レベル,最も細かいメッシュ系のレベル=1とする)を予め準備しておき,各メッシュ幅(格子幅)に対応した波長の残差成分を減衰させる方法である。
- 各波長の残差成分を一様に減衰できるため,問題規模によらず反復回数が一様なアルゴリズムを構成できる。
  - 1,000倍の規模の問題を解くのに1,000倍の数のCPUを用意すれば,同じ時間で計算ができる
- 弱スケーリング(Weak Scaling)
  - 1CPU当たりの問題規模を固定して,CPU数を増やす
  - 1,000倍の規模の計算を,CPU数を1,000倍にして解くと:
    - CG法(ICCG法)では反復回数・計算時間ともに増加
    - 多重格子法(Multigrid)では反復回数・計算時間ともに一定:スケーラブル(Scalable)

弱スケーリングの例（Weak Scaling）
三次元ポアソン方程式 ($\Delta\phi=q$)

ICCG = 不完全コレスキー分解前処理＋共役勾配法（CG）

MGCG= マルチグリッド前処理（MG）＋共役勾配法（CG）



DOF：Degrees of Freedom
自由度，未知数の数

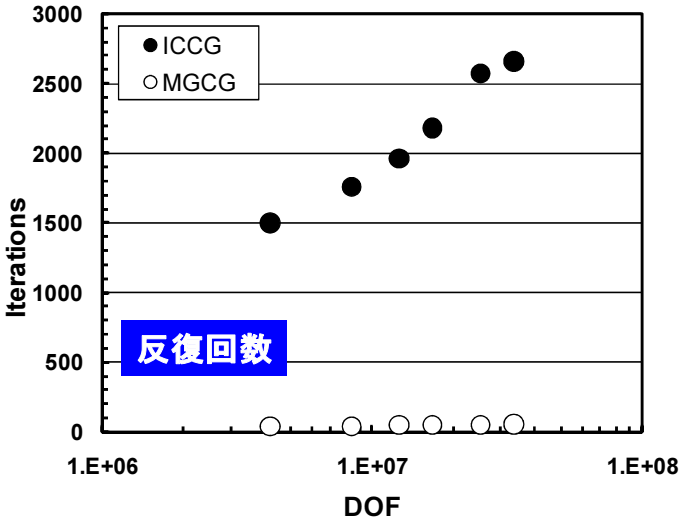
弱スケーリングの例

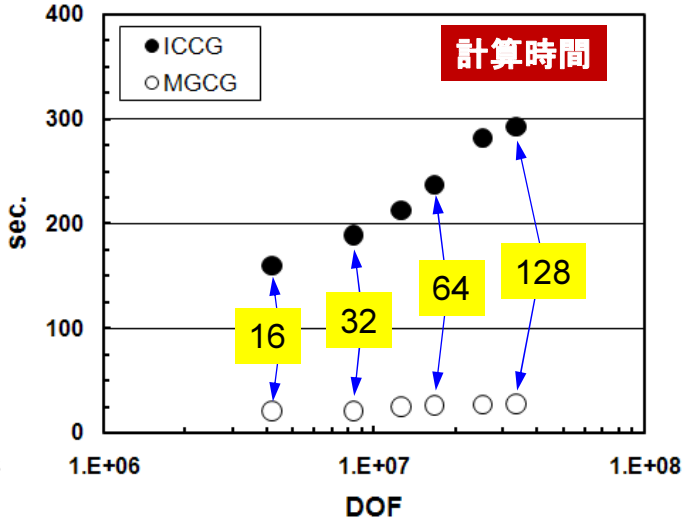
プロセッサ数増加⇒問題規模増加

三次元ポアソン方程式 ($\Delta\phi=q$)

ICCG：反復回数増加⇒計算時間増加

MGCG：反復回数，計算時間不変⇒スケーラブル（Scalable）



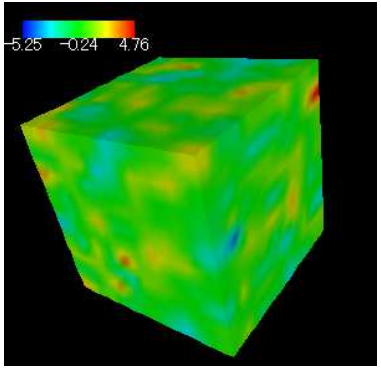


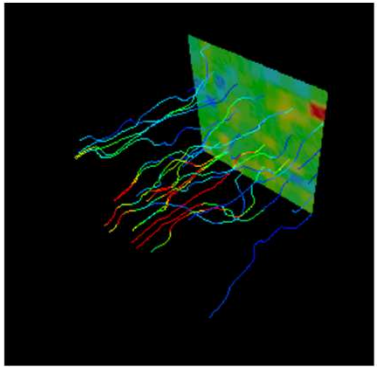
並列多重格子法:Parallel Multigrid

- 不均質多孔質媒体(堆積岩)中の地下水流れ
  - Darcy流れ
  - (λ=10^{-5}-10^{+5}):透水係数(水の通しやすさ:穴の数)
  - 有限堆積法(Finite Volume Method)
  - MGCG法(MG法+CG法)

$$\nabla \cdot \left(\lambda(x,y,z) \nabla \phi \right) = q$$

$$u = -\lambda \frac{\partial \phi}{\partial x}, v = -\lambda \frac{\partial \phi}{\partial y}, w = -\lambda \frac{\partial \phi}{\partial z}$$

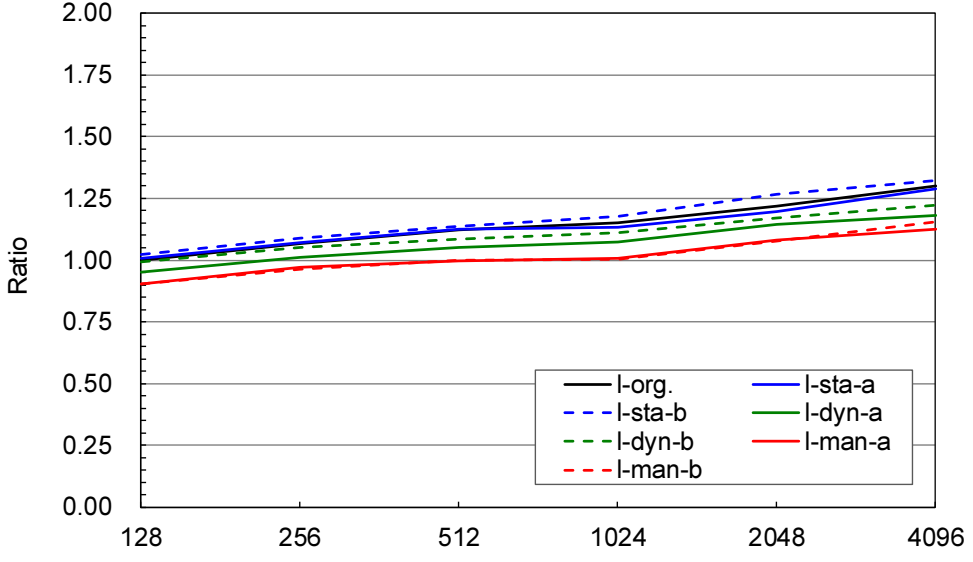




- Wisteria/BDEC-01(Odyssey)
  - 最大4,096ノード
  - 最大問題規模: 34,359,738,368(343億自由度)

Weak Scalingの結果（128~4,096ノード）：Odyssey 128ノードの計算時間を1.00として無次元化

いくつかの手法を比較しているのだが，ここでは，32倍の規模の問題を解いても計算時間があまり変わっていないことに注目してほしい



Weak Scalingの結果（128~4,096ノード）：Odyssey

128ノー[illegible]

多重格子法は解ける問題が限定されるのが難であったが，より広範囲の問題を解くための研究が進められている。

いくつかの[illegible]を解いても計算時[illegible]い