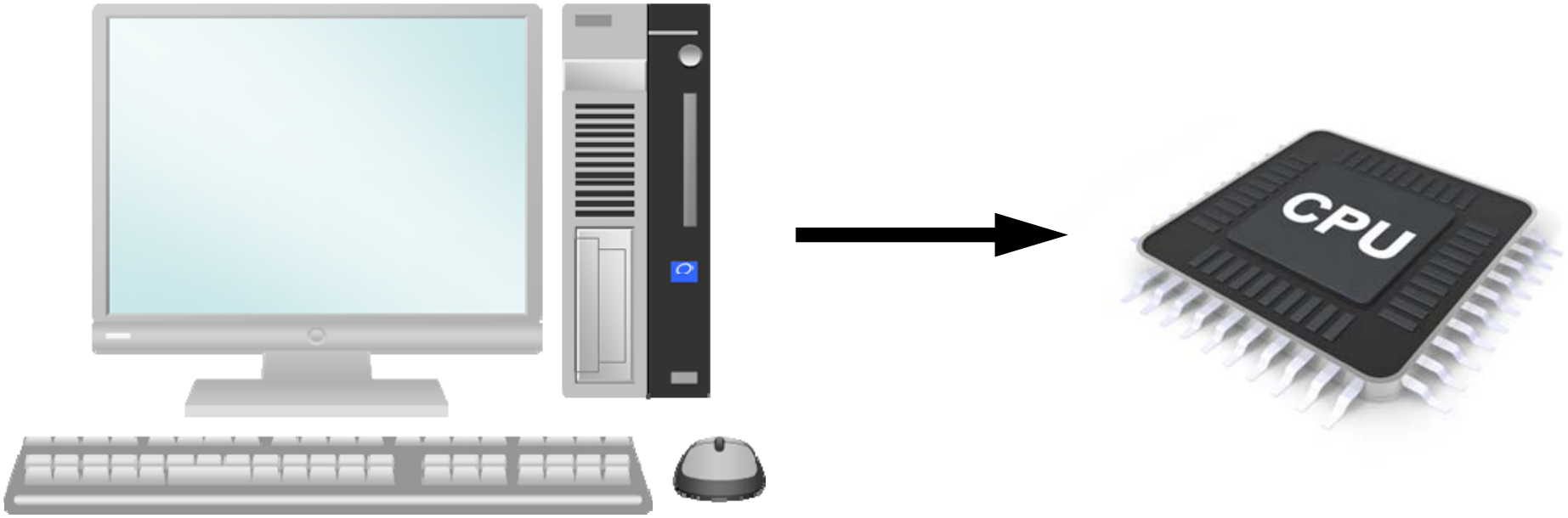
スーパーコンピューティングへの招待
プログラミングについて

http://nkl.cc.u-tokyo.ac.jp/16n/

中島 研吾
東京大学情報基盤センター
同　大学院情報理工学系研究科数理情報学専攻
同　大学院工学系研究科電気系工学専攻

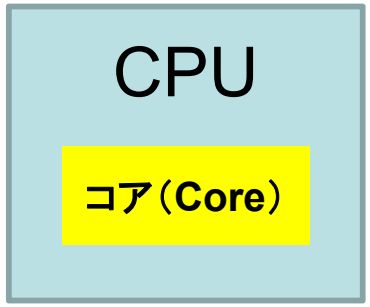
- スーパーコンピュータ(スパコン)とは
- スパコンによる科学技術シミュレーション
- SMASH
- プログラミングについて

計算機とCPU

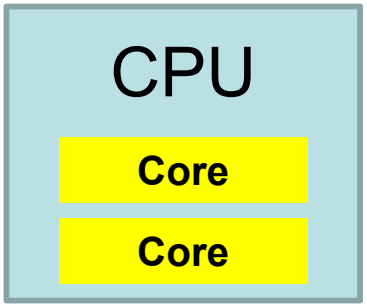


- Central Processing Unit（中央処理装置）: CPU
- PCで利用されているCPUとスーパーコンピュータ（スパコン）のCPUは基本的に同じ構造（Architecture：アーキテクチャ）
- GHz: Clock Rate（クロック数）
  - Frequency（振動数）: 1秒間にCPUが動作を実行する回数
    - GHz -> 10^9 回
  - 1クロックに4-8命令を同時実行

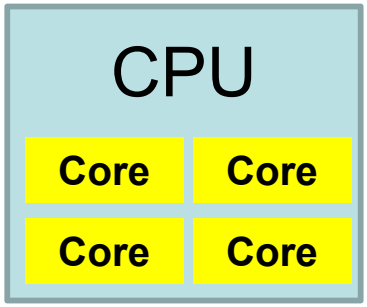
マルチコアCPU



Single Core
1 cores/CPU



Dual Core
2 cores/CPU



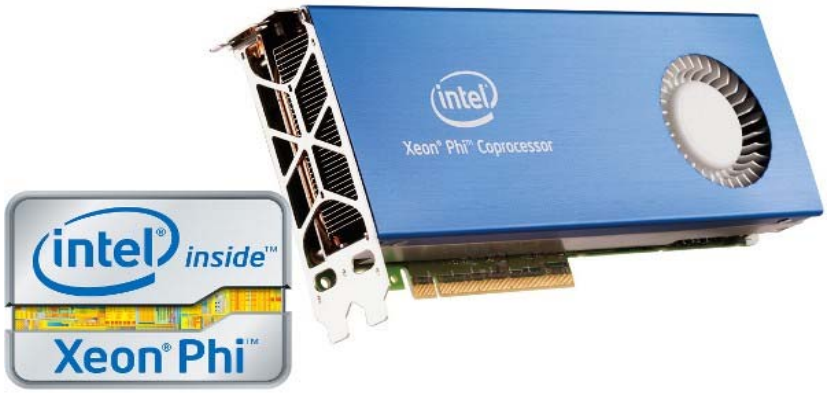
Quad Core
4 cores/CPU

- コア（Core）= CPUの主要計算部分
- 4-8コアから成るマルチコアCPUが一般的

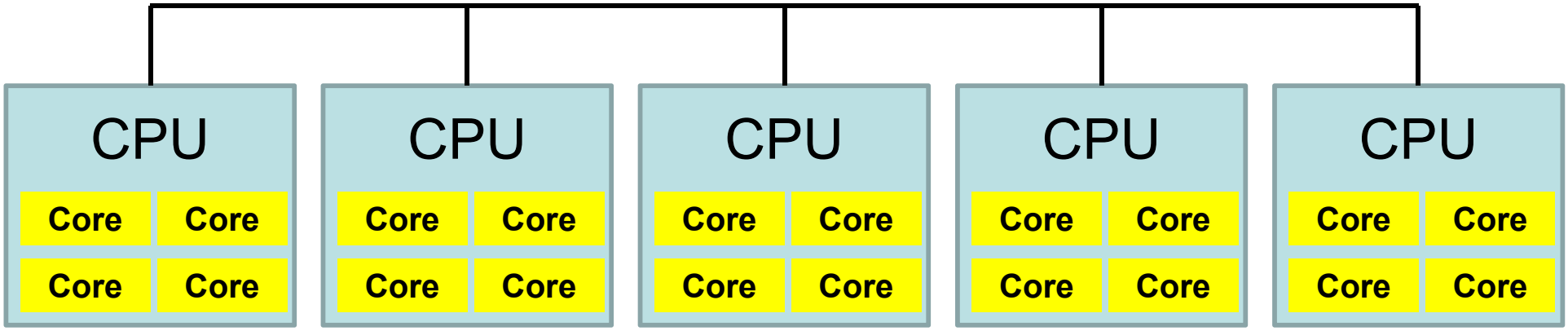
- GPU：メニーコア（Manycore）
  - $O(10^1)$-$O(10^2)$ cores
- 多くのコアを使った並列計算（Parallel Computing）
- 東大Oakleaf-FX: 16 cores
  - SPARC64™ IXfx

GPU/Manycores

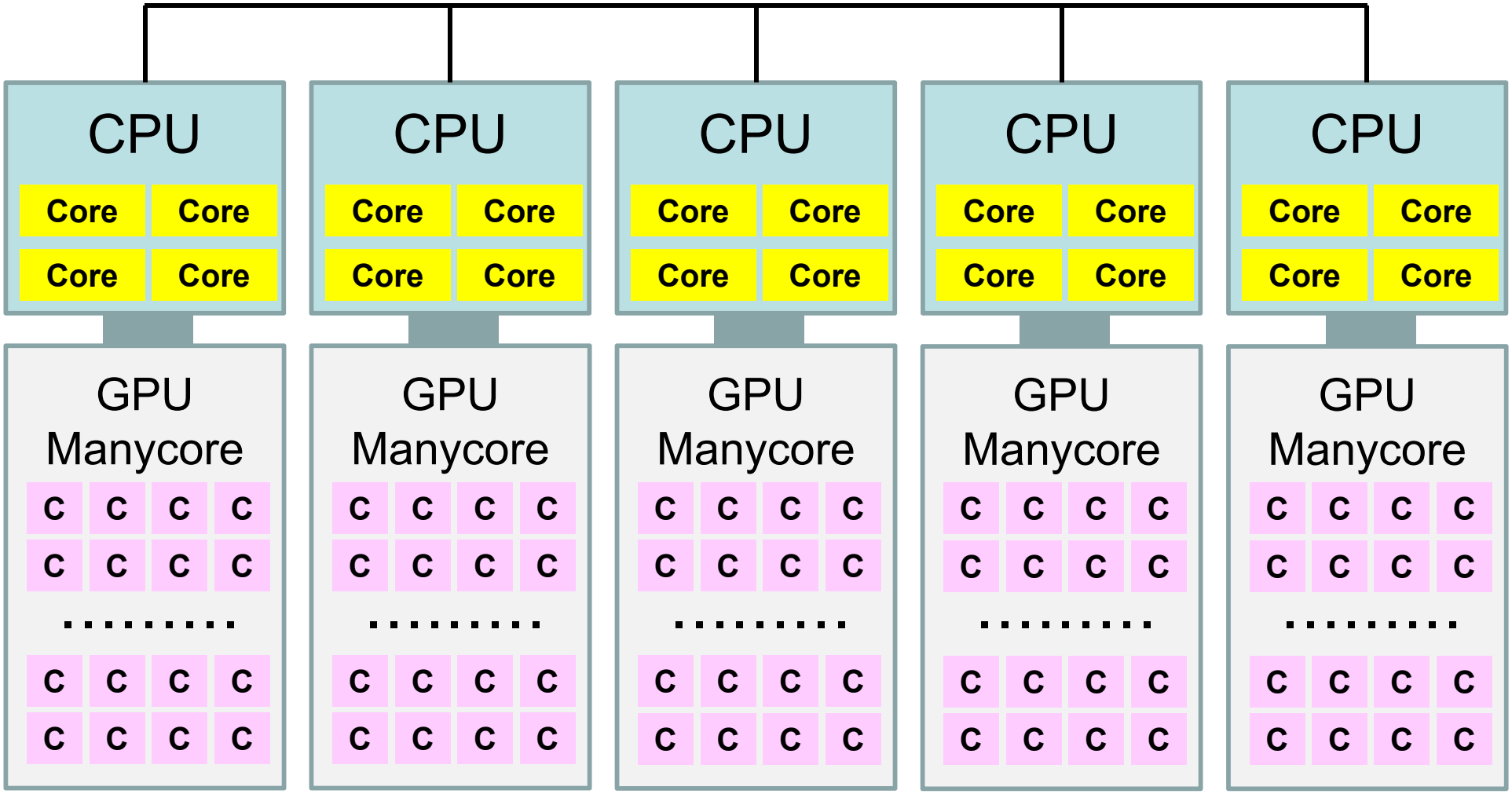
- GPU：Graphic Processing Unit
  - 本来画像処理用途
  - GPGPU: General Purpose GPU
  - O(10^2) cores
  - 高メモリバンド幅（Memory Bandwidth）
  - 廉い
  - ホストCPUが必要
  - Programming: CUDA, OpenACC
- Intel Xeon/Phi: Manycore CPU
  - 60 cores
  - 高メモリバンド幅
  - Unix, Fortran, C compilerが動く
  - 現状ではホストCPUが必要
    - 最新機種では不要



並列スーパーコンピュータ（スパコン）
Parallel Supercomputers
マルチコアCPUをネットワーク経由で接続



ヘテロジニアス・ハイブリッドな計算ノードを有する並列スーパーコンピュータ
Heterogeneous/Hybrid Nodes



スパコンの性能

- CPUの性能:クロック数で決まる
- FLOPS (Floating Point Operations per Second)
  - 1秒当たりの浮動小数点(実数)演算回数
- 昨今のマルチコアCPU
  - 4-8 FLOPS per Clock
  - (例)3GHzマルチコアCPUのピーク性能(物理最大性能)
    - $3 \times 10^9 \times 4(\text{or } 8) = 12(\text{or } 24) \times 10^9$ FLOPS=12(or 24)GFLOPS

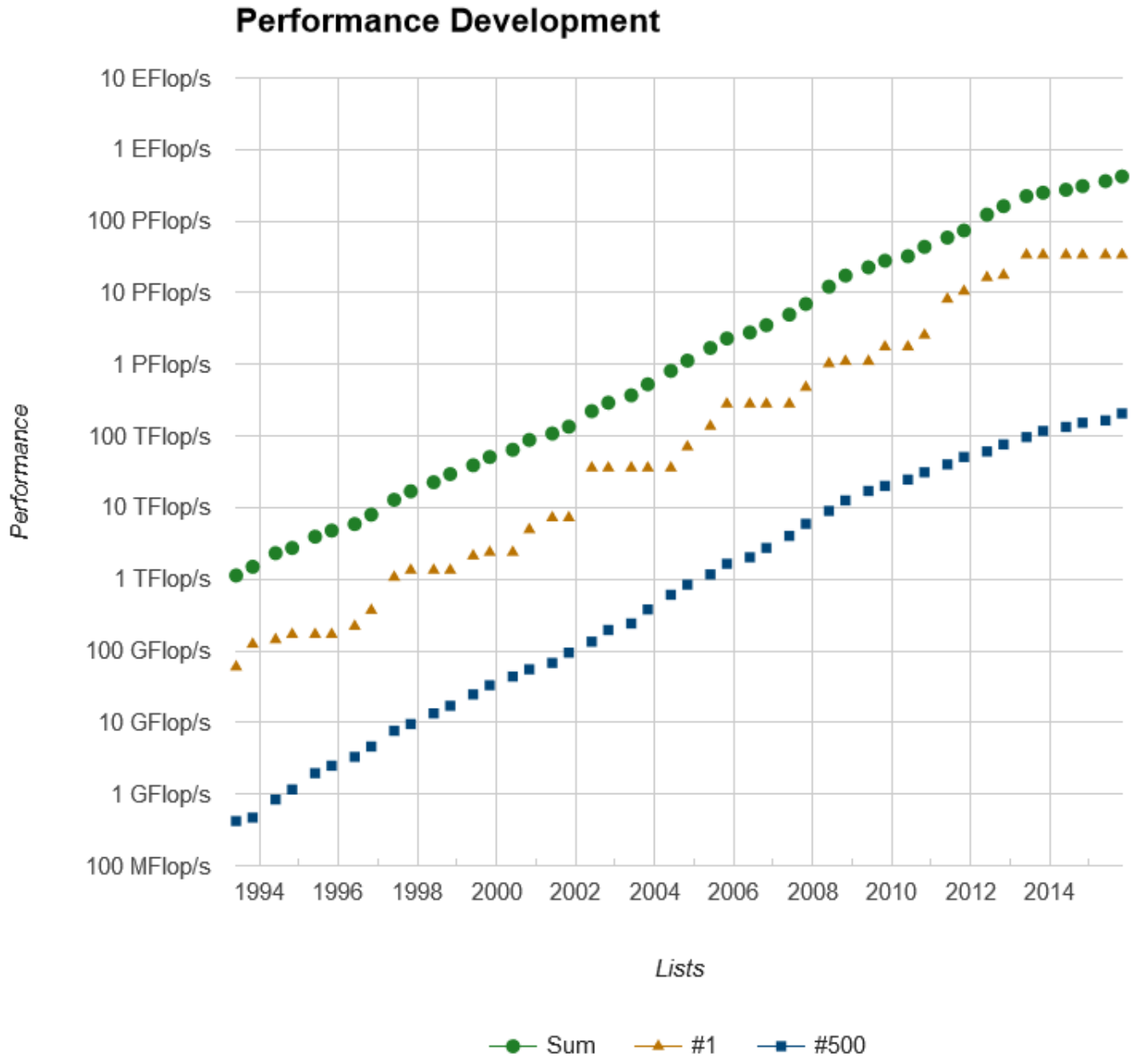
    - 10^6 FLOPS= 1 Mega FLOPS = 1 MFLOPS
    - 10^9 FLOPS= 1 Giga FLOPS = 1 GFLOPS
    - 10^{12} FLOPS= 1 Tera FLOPS = 1 TFLOPS
    - 10^{15} FLOPS= 1 Peta FLOPS = 1 PFLOPS
    - 10^{18} FLOPS= 1 Exa FLOPS = 1 EFLOPS

TOP 500 List

http://www.top500.org/

- 年2回更新（1993年～）
- LINPACKと言われるベンチマークテストを実施し，FLOPS値を測定する
  - 密行列を係数とする連立一次方程式を解く
  - iPhone/iPad版もある

- 実際のアプリケーションではこれほどの性能は出ない
  - 実アプリケーションに即したベンチマークを使用する提案もある（HPCG）



- PFLOPS: Peta (=10^{15}) Floating OPerations per Sec.
- Exa-FLOPS (=10^{18}) will be attained after 2023 …

http://www.top500.org/

47th TOP500 List (June, 2016)

	Site	Computer/Year Vendor	Cores	R_{max}	R_{peak}	Power
1	**National Supercomputing Center in Wuxi, China**	**Sunway TaihuLight - Sunway MPP, Sunway SW26010 260C 1.45GHz, 2016 NRCPC**	**10649600**	**93015 (= 93.0 PF)**	**125436**	**15371**
2	National Supercomputing Center in Tianjin, China	**Tianhe-2** Intel Xeon E5-2692, TH Express-2, Xeon Phi, 2013 NUDT	3120000	33863 (= 33.9 PF)	54902	17808
3	Oak Ridge National Laboratory, USA	**Titan** Cray XK7/NVIDIA K20x, 2012 Cray	560640	17590	27113	8209
4	Lawrence Livermore National Laboratory, USA	**Sequoia** BlueGene/Q, 2011 IBM	1572864	17173	20133	7890
5	**RIKEN AICS, Japan**	**K computer, SPARC64 VIIIfx , 2011 Fujitsu**	**705024**	**10510**	**11280**	**12660**
6	Argonne National Laboratory, USA	**Mira** BlueGene/Q, 2012 IBM	786432	8587	10066	3945
7	**DOE/NNSA/LANL/SNL, USA**	**Trinity - Cray XC40, Xeon E5-2698v3 16C 2.3GHz, 2016 Cray**	**301056**	**8101**	**11079**	**3945**
8	Swiss Natl. Supercomputer Center, Switzerland	**Piz Daint** Cray XC30/NVIDIA K20x, 2013 Cray	115984	6271	7789	2325
9	**HLRS-Stuttgart, Germany**	**Hazel Hen - Cray XC40, Xeon E5-2680v3 12C 2.5GHz, 2016 Cray**	**185088**	**5640**	**7404**	**-**
10	KAUST, Saudi Arabia	**Shaheen II** Cray XC40, Xeon E5-2698, 2015 Cray	196608	5537	7235	2834

R_{max}: Performance of Linpack (TFLOPS)
R_{peak}: Peak Performance (TFLOPS), Power: kW

http://www.top500.org/

47th TOP500 List (June, 2016)

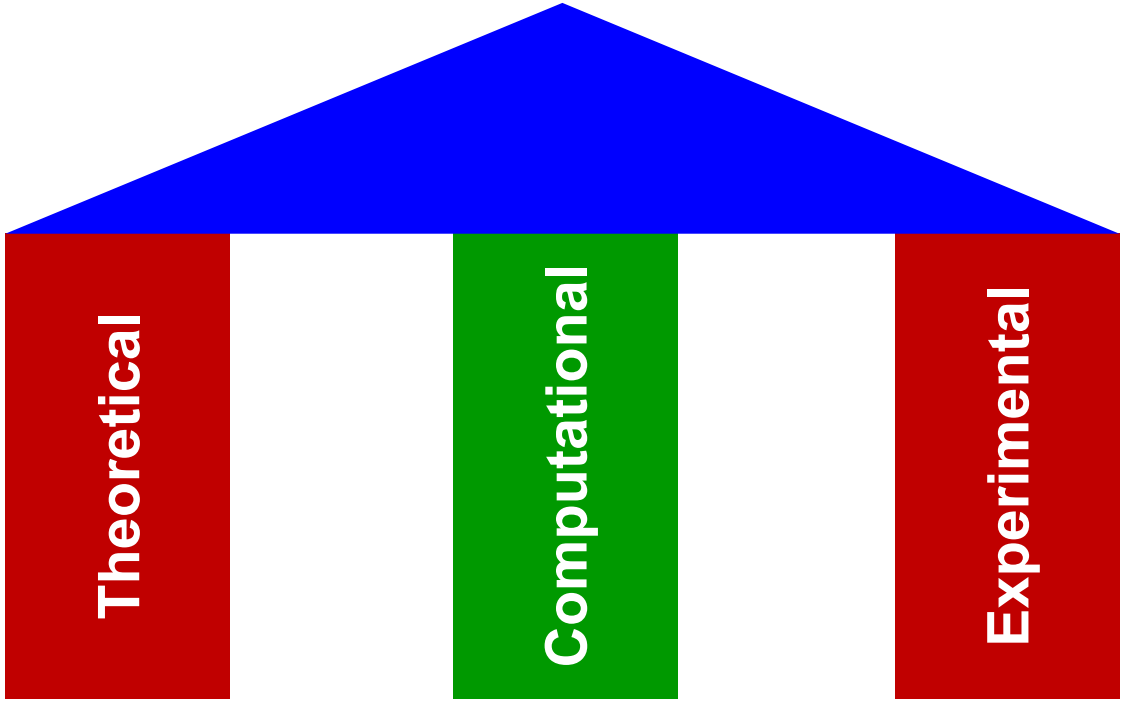
	Site	Computer/Year Vendor	Cores	R_{max}	R_{peak}	Power
1	National Supercomputing Center in Wuxi, China	Sunway TaihuLight - Sunway MPP, Sunway SW26010 260C 1.45GHz, 2016 NRCPC	10649600	93015 (= 93.0 PF)	125436	15371
2	National Supercomputing Center in Tianjin, China	**Tianhe-2** Intel Xeon E5-2692, TH Express-2, Xeon Phi, 2013 NUDT	3120000	33863 (= 33.9 PF)	54902	17808
3	Oak Ridge National Laboratory, USA	**Titan** Cray XK7/NVIDIA K20x, 2012 Cray	560640	17590	27113	8209
4	Lawrence Livermore National Laboratory, USA	**Sequoia** BlueGene/Q, 2011 IBM	1572864	17173	20133	7890
5	RIKEN AICS, Japan	K computer, SPARC64 VIIIfx , 2011 Fujitsu	705024	10510	11280	12660
6	Argonne National Laboratory, USA	**Mira** BlueGene/Q, 2012 IBM	786432	8587	10066	3945
7	DOE/NNSA/LANL/SNL, USA	Trinity - Cray XC40, Xeon E5-2698v3 16C 2.3GHz, 2016 Cray	301056	8101	11079	3945
8	Swiss Natl. Supercomputer Center, Switzerland	**Piz Daint** Cray XC30/NVIDIA K20x, 2013 Cray	115984	6271	7789	2325
9	HLRS-Stuttgart, Germany	Hazel Hen - Cray XC40, Xeon E5-2680v3 12C 2.5GHz, 2016 Cray	185088	5640	7404	-
10	KAUST, Saudi Arabia	**Shaheen II** Cray XC40, Xeon E5-2698, 2015 Cray	196608	5537	7235	2834
85	ITC/U. Tokyo Japan	Oakleaf-FX SPARC64 IXfx, 2012 Fujitsu	76800	1043	1135	1177

http://www.top500.org/

- スーパーコンピュータ(スパコン)とは
- **スパコンによる科学技術シミュレーション**
- SMASH
- プログラミングについて

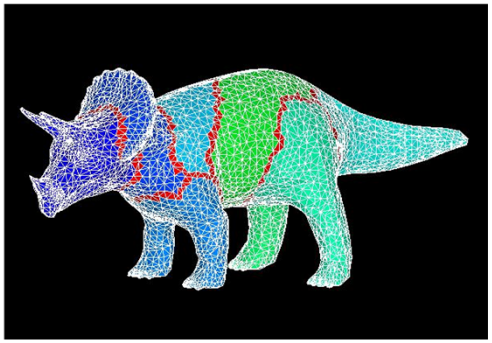
計算科学：Computational Science
自然科学の「第3の柱」：The 3rd Pillar of Science

- 理論：Theoretical，実験：Experimental
- 計算科学：Computational Science
  - The 3rd Pillar of Science
  - スパコンを駆使した計算機シミュレーション，科学技術シミュレーション

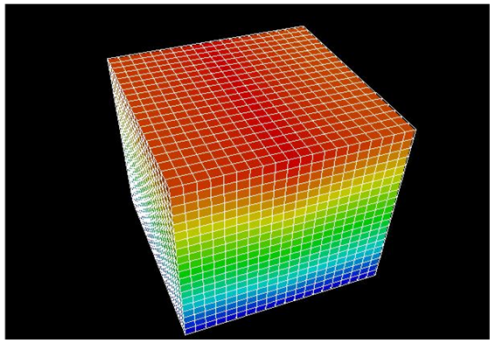


科学技術計算の方法

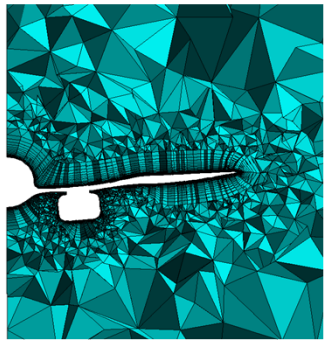
- 偏微分方程式（Partial Differential Equations：PDE）数値解
- メッシュ，格子，粒子（mesh, grid, particle）
  - 大規模な連立一次方程式を解く必要あり
  - 細かいメッシュほど計算量は多いが精度の良い解



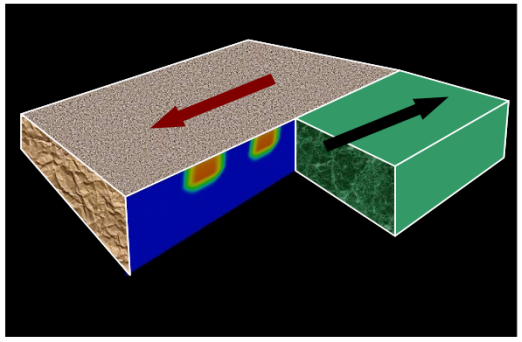
有限要素法
Finite Element Method
FEM



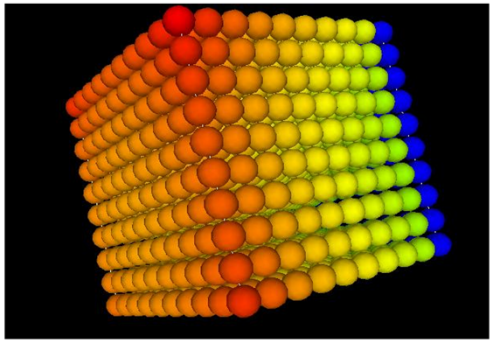
差分法
Finite Difference Method
FDM



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

大規模連立一次方程式Ax=bを解く！

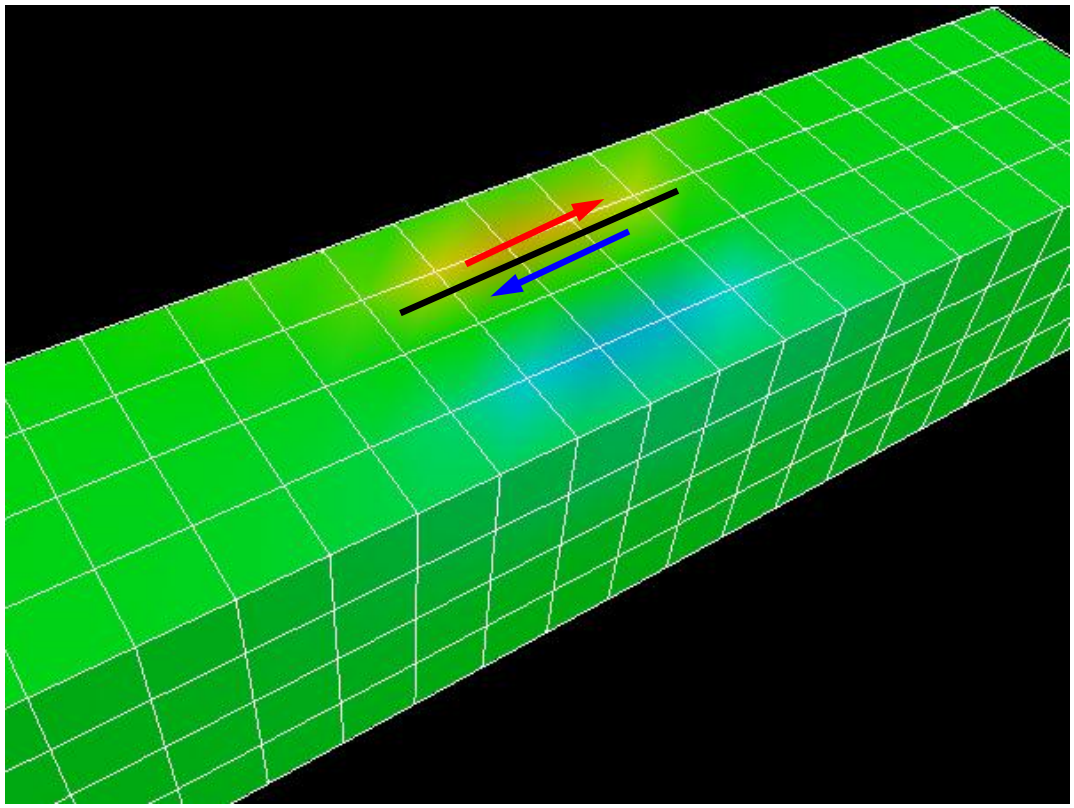
- （教養の線形代数学（数学Ⅱ）（昔の「幾何」））
- 科学技術シミュレーションにおいて最も重要なプロセス
  - 時間がかかる
  - 安定で高速な手法が必須：新しい科学の開拓に貢献
- 解法
  - 直接法（ガウス消去法，LU分解）
  - 反復法（定常法，非定常法（クリロフ（Krylov）部分空間法））
- 係数行列A
  - 密行列，疎行列（ゼロ多い）：モデル化手法，条件により異なる
  - 行列を見れば現象の全てがわかる！
- 研究としてやることはたくさんある，実用性も高い
  - 詳しくは11月以降に

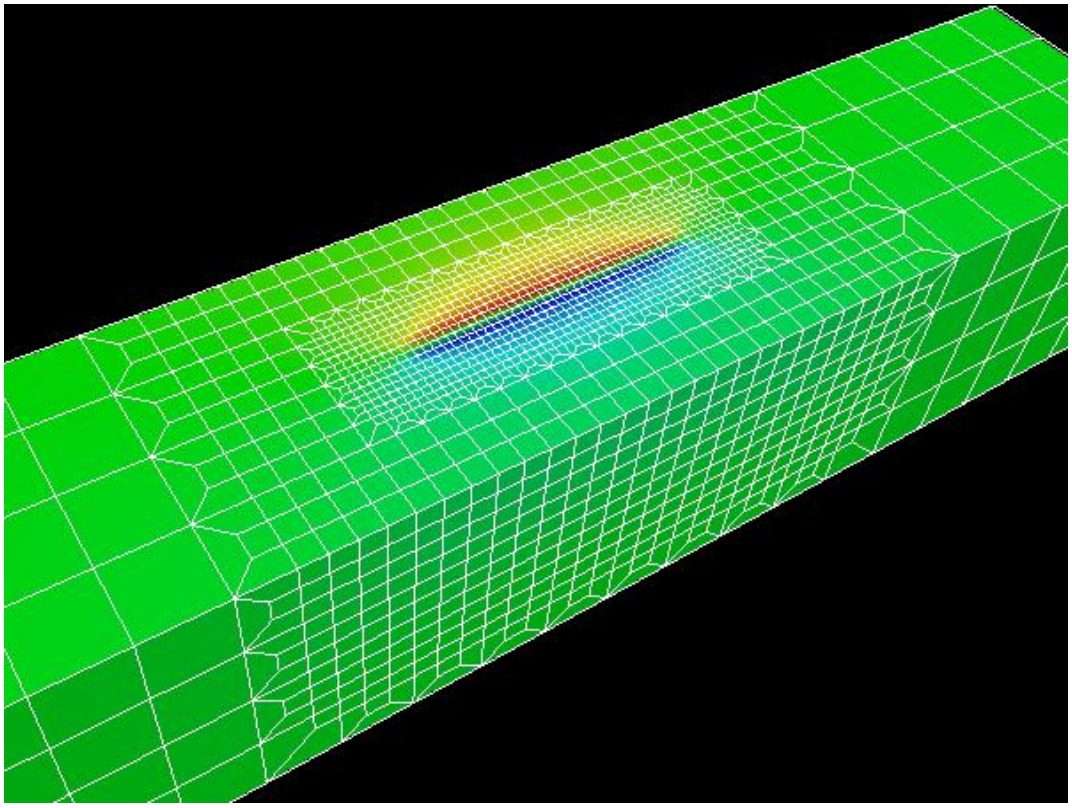
地震発生メカニズム研究：有限要素法の例

San Andreas Faults, CA, USA

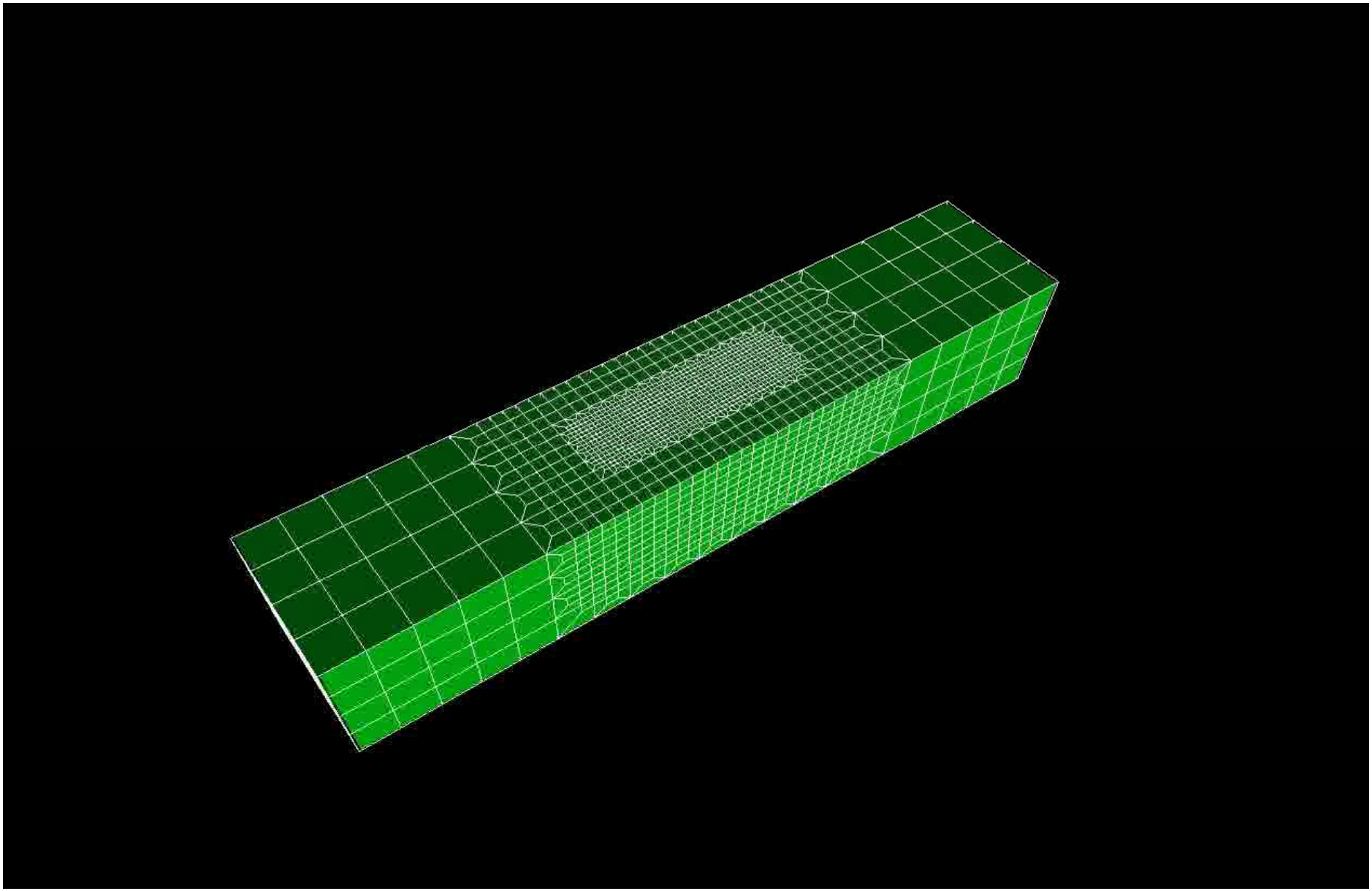
Stress Accumulation at Transcurrent Plate Boundaries

地震を引き起こす力が蓄積する断層周辺には
高解像度の細かいメッシュを適用している





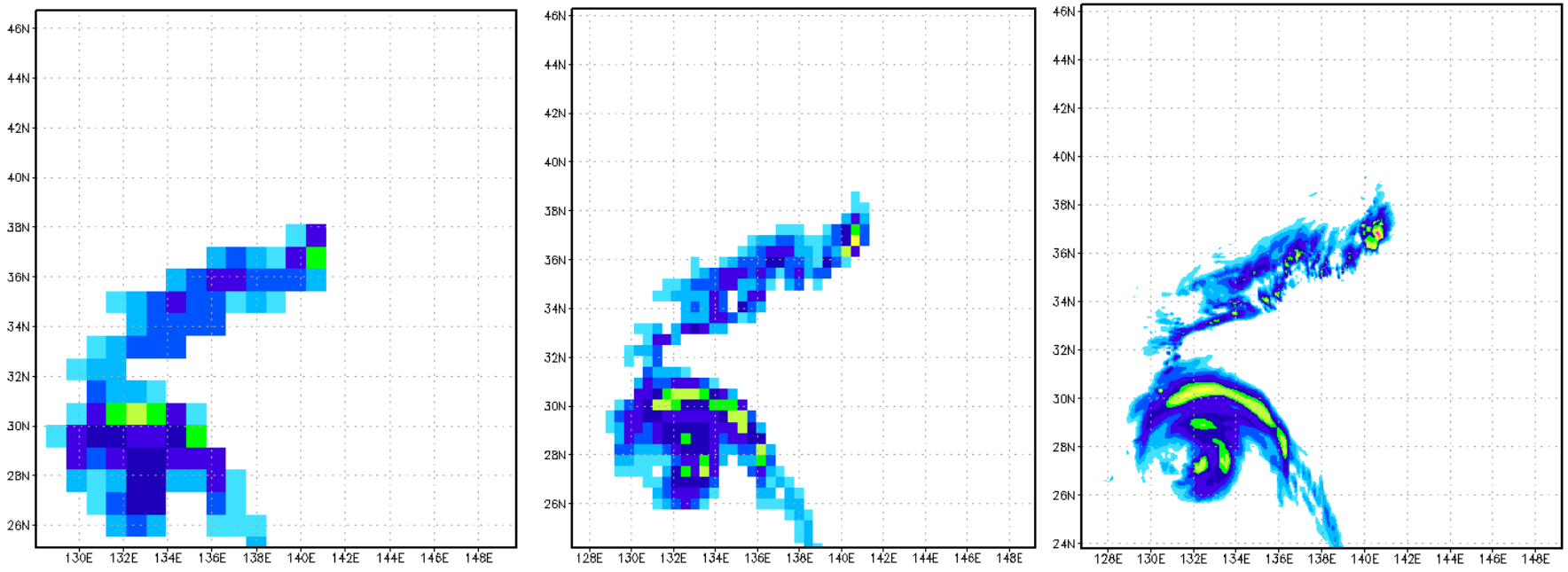
変化量・力の蓄積の大きい断層周辺では高解像度の細かいメッシュが必要



差分法による台風のシミュレーション
解像度の影響

低解像度

高解像度



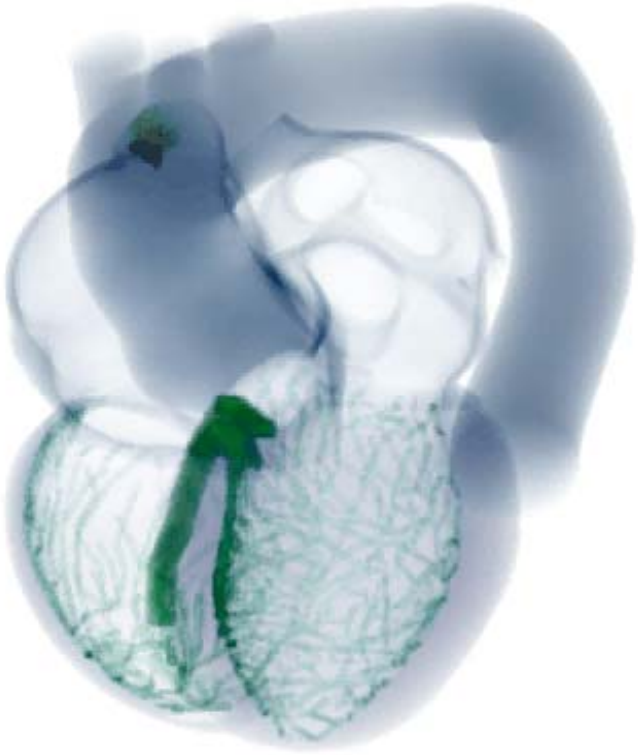
100kmメッシュ

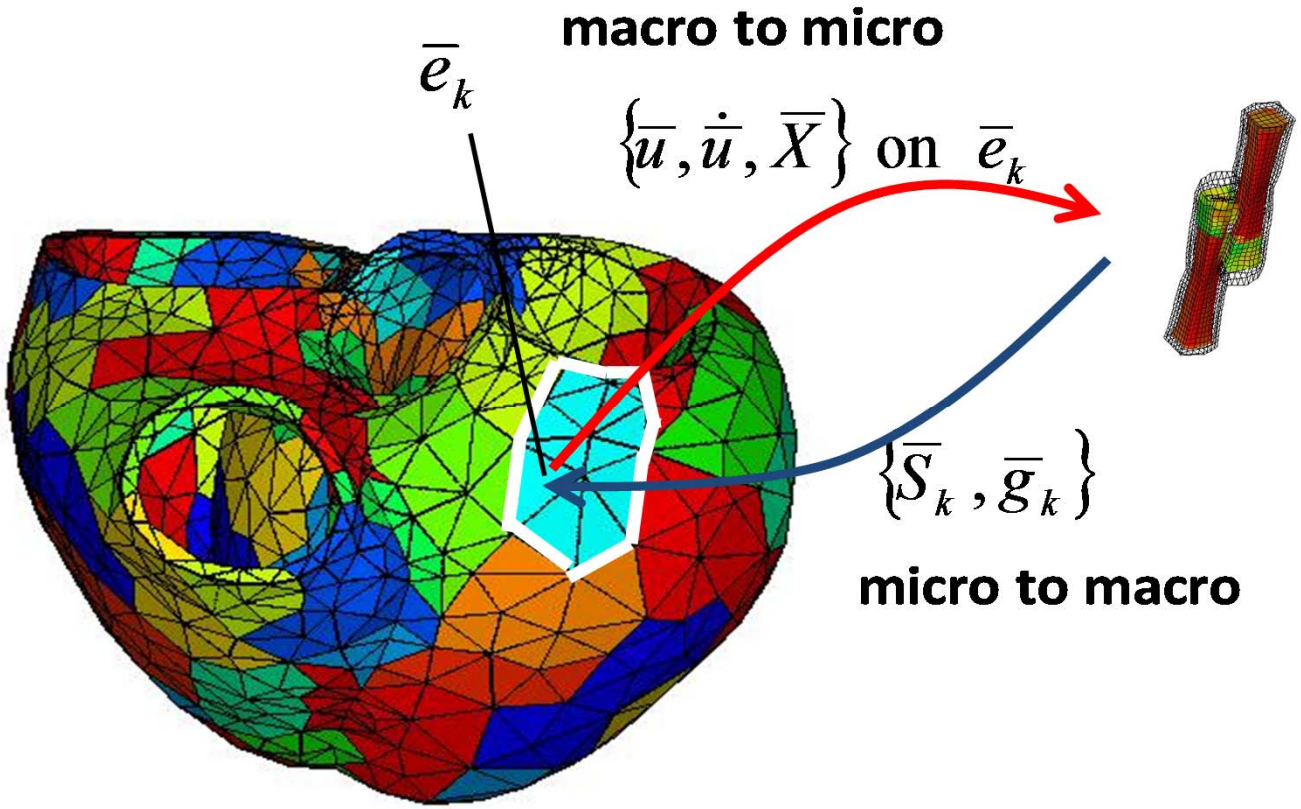
50kmメッシュ

5kmメッシュ

〔画像提供：(独)海洋研究開発機構〕

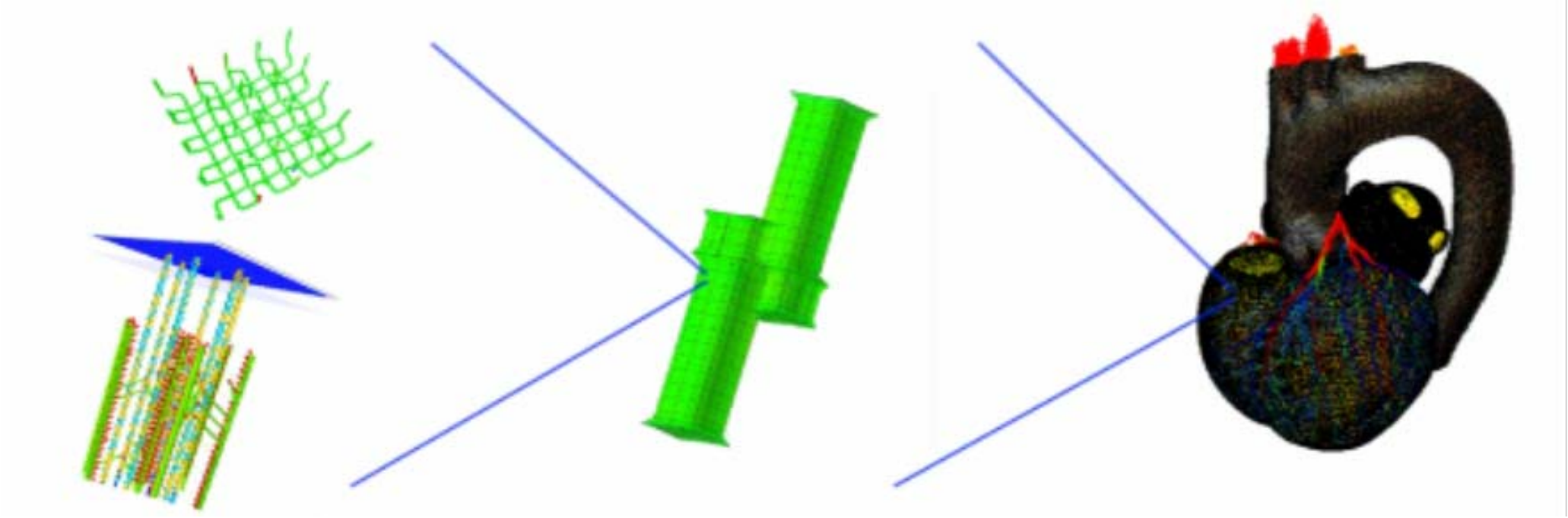
「京」コンピュータによるマルチスケール心臓シミュレーション





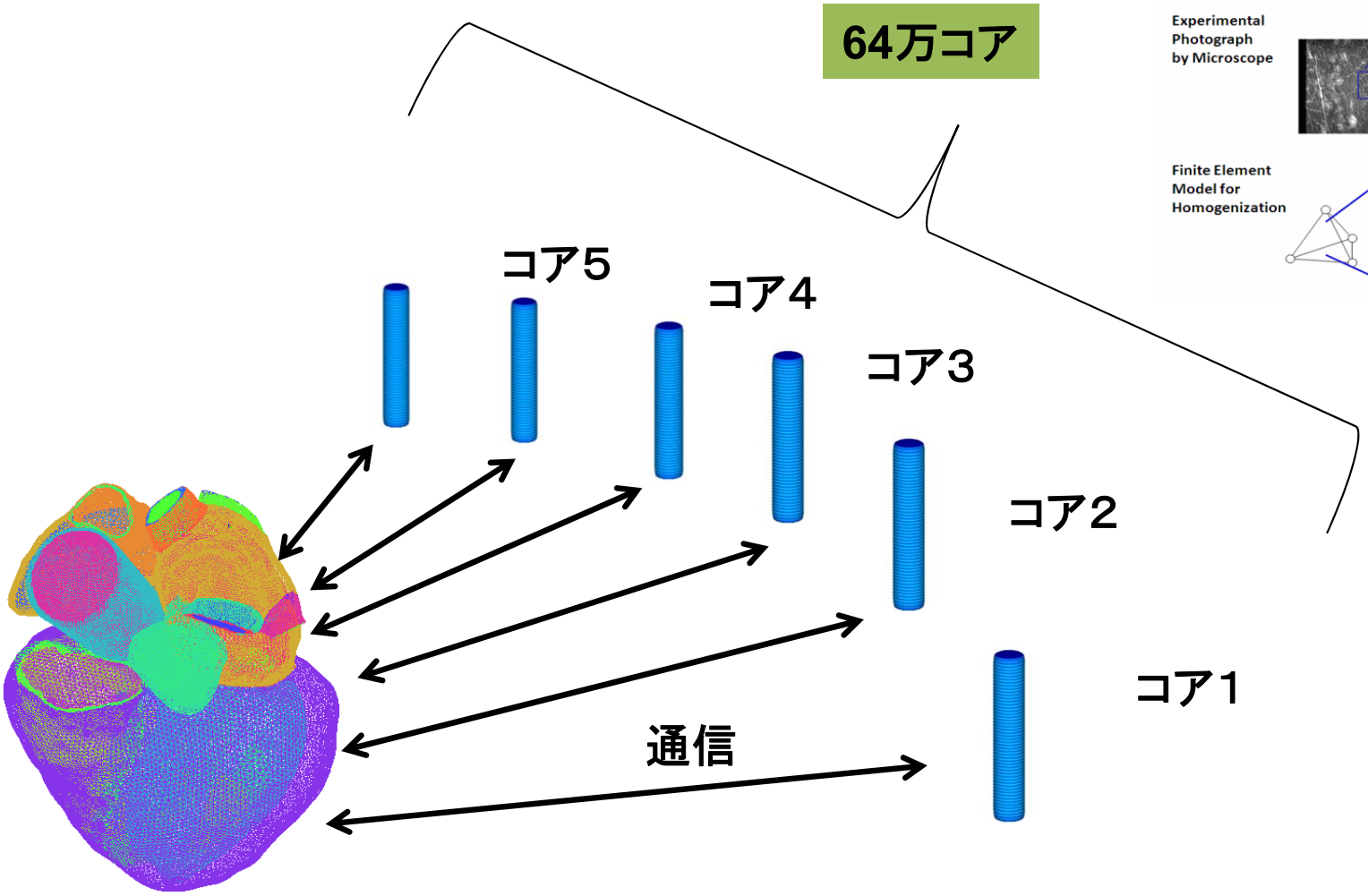
〔資料提供：東京大学　久田・杉浦・鷲尾・岡田研究室，富士通（株）〕

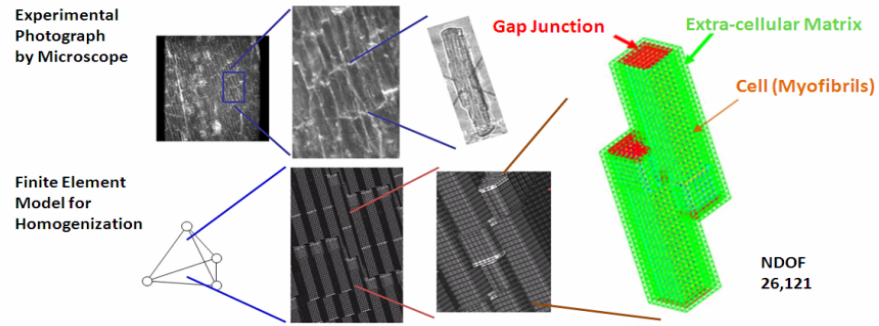
マルチスケール心臓シミュレーション



〔資料提供：東京大学　久田・杉浦・鷲尾・岡田研究室，富士通（株）〕

「京」コンピュータによるマルチスケール心臓シミュレーション　概念図



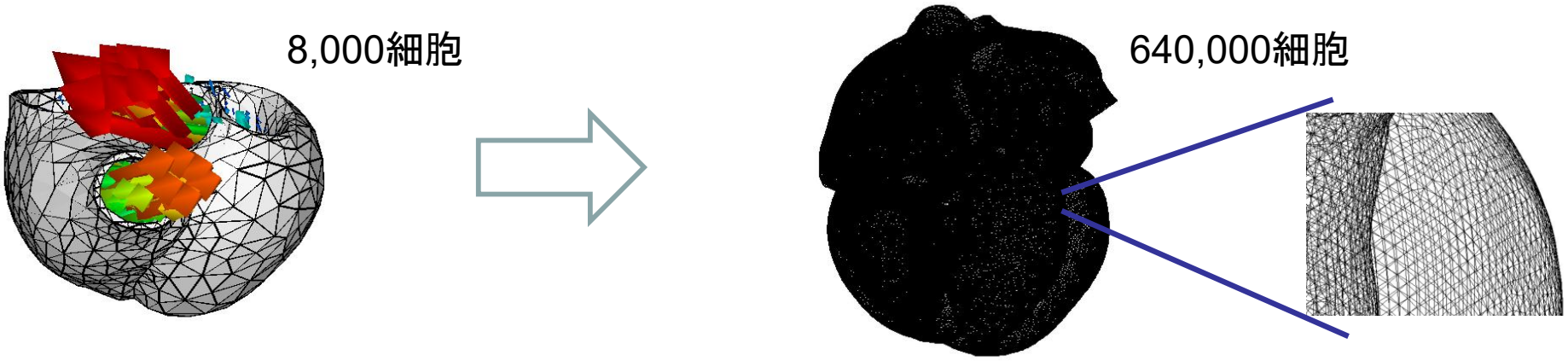


細胞間の直接的通信は発生しない

〔資料提供：東京大学　久田・杉浦・鷲尾・岡田研究室，富士通（株）〕

「京」コンピュータによって・・・

- 生理学的に正しい心臓拍動のシミュレーションが可能
  - 心臓では実際は興奮波が心筋壁内を伝播する。
  - T2K東大（8,000コア）では再現できないが，京コンピュータでは再現できる
- 京コンピュータ（全系）を使っても1心拍の計算には2日程度の計算時間がかかる
  - 20万自由度細胞 x 64万個 = 1,280億自由度



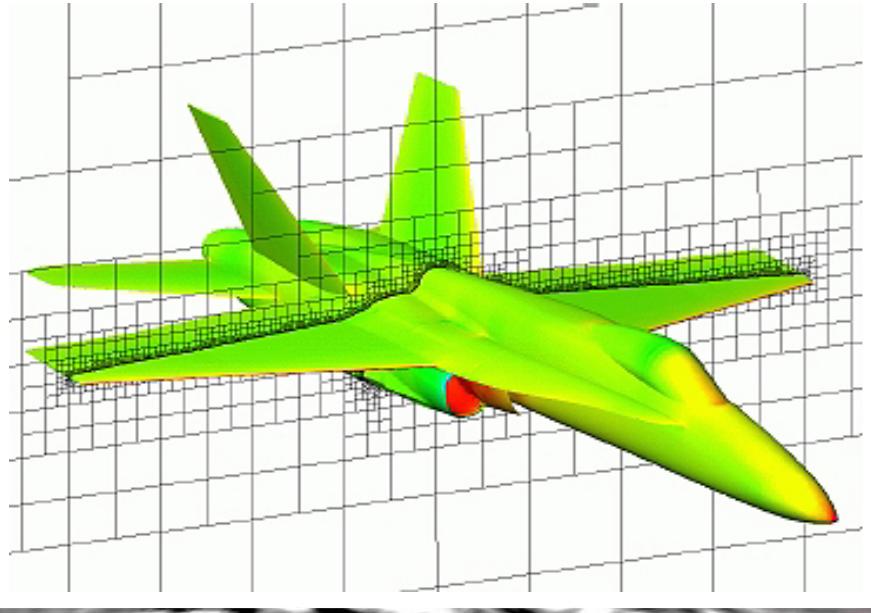
T2K東大 8,000コア

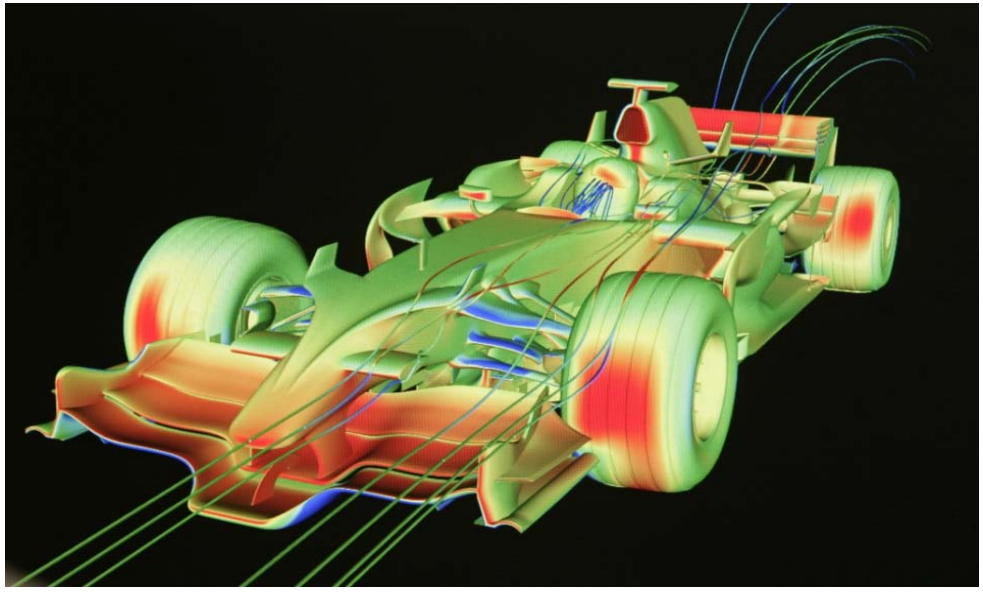
京コンピュータ　64万コア

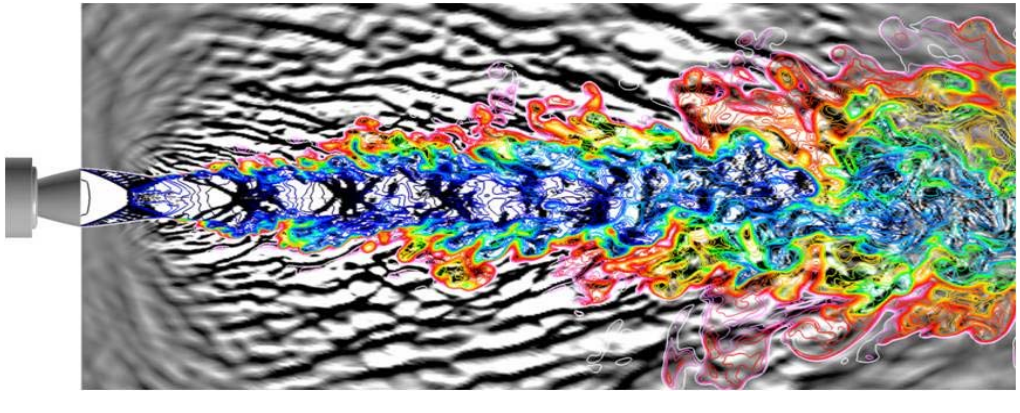
〔資料提供：東京大学　久田・杉浦・鷲尾・岡田研究室，富士通（株）〕

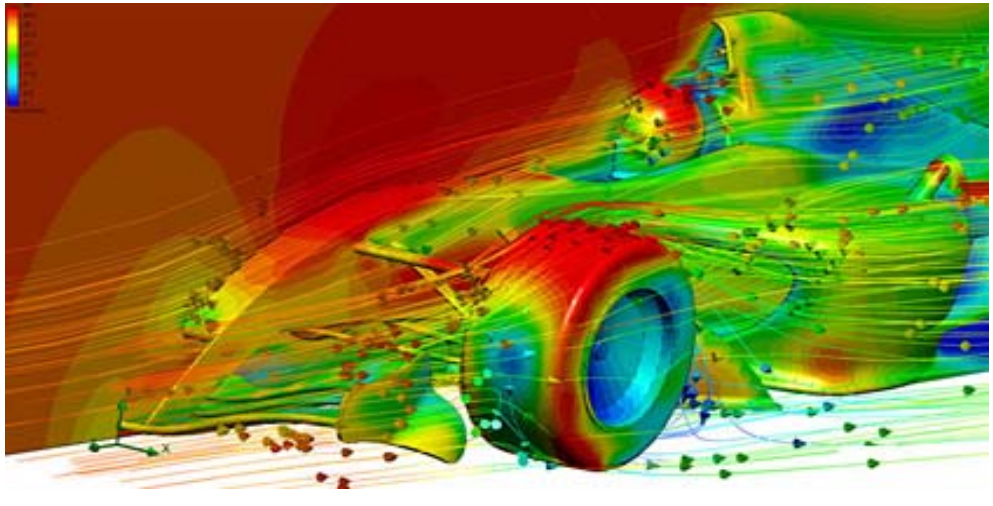
数値流体力学
（Computational Fluid Dynamics）

- 航空機，自動車周囲・エンジン内空気の流れ計算→空気抵抗軽減，省エネにつながる



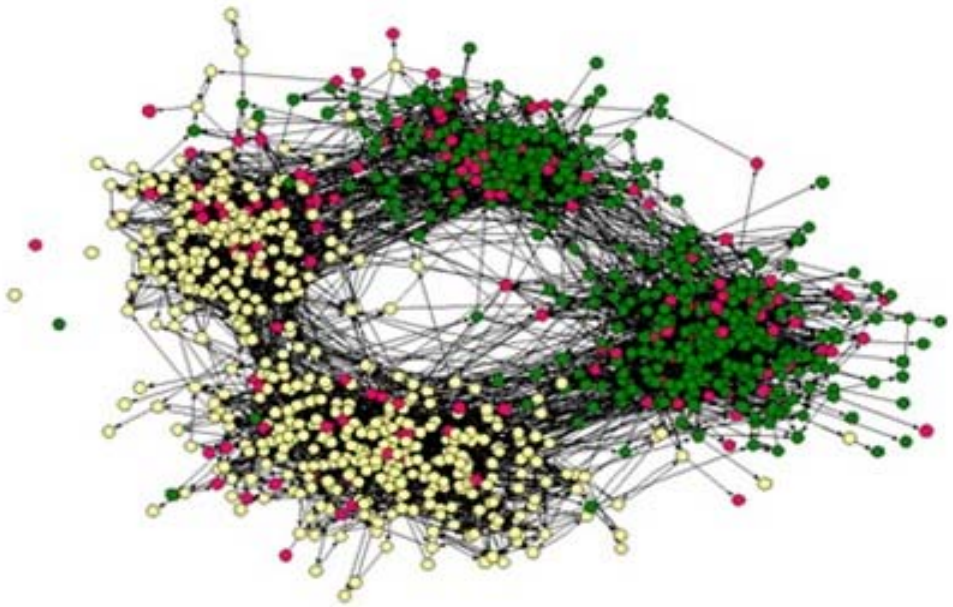


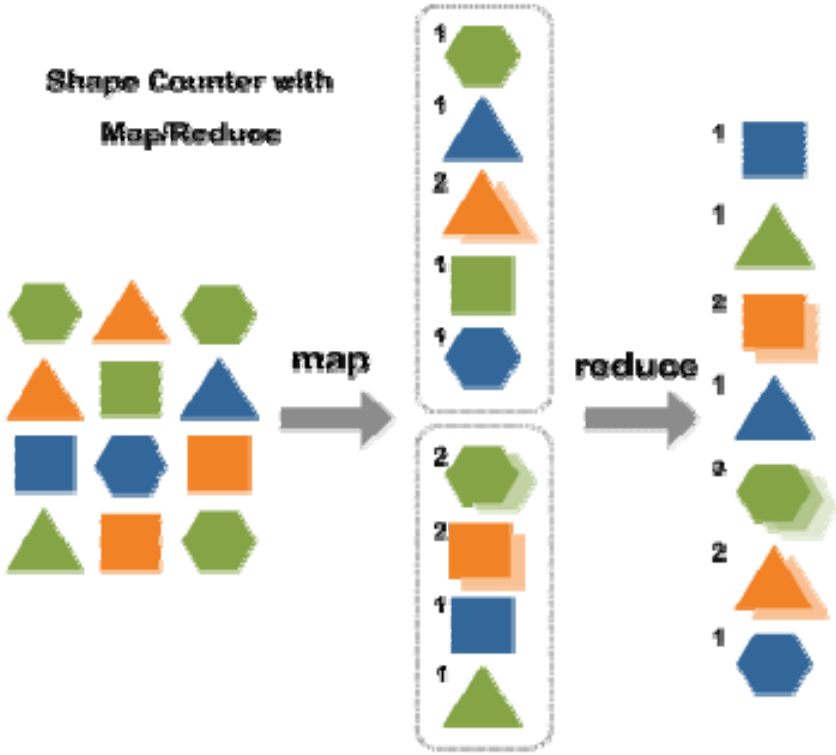




大規模データ処理

- ゲノムの解読
- 衛星観測データ
- 大規模テキスト処理
  - Google　等

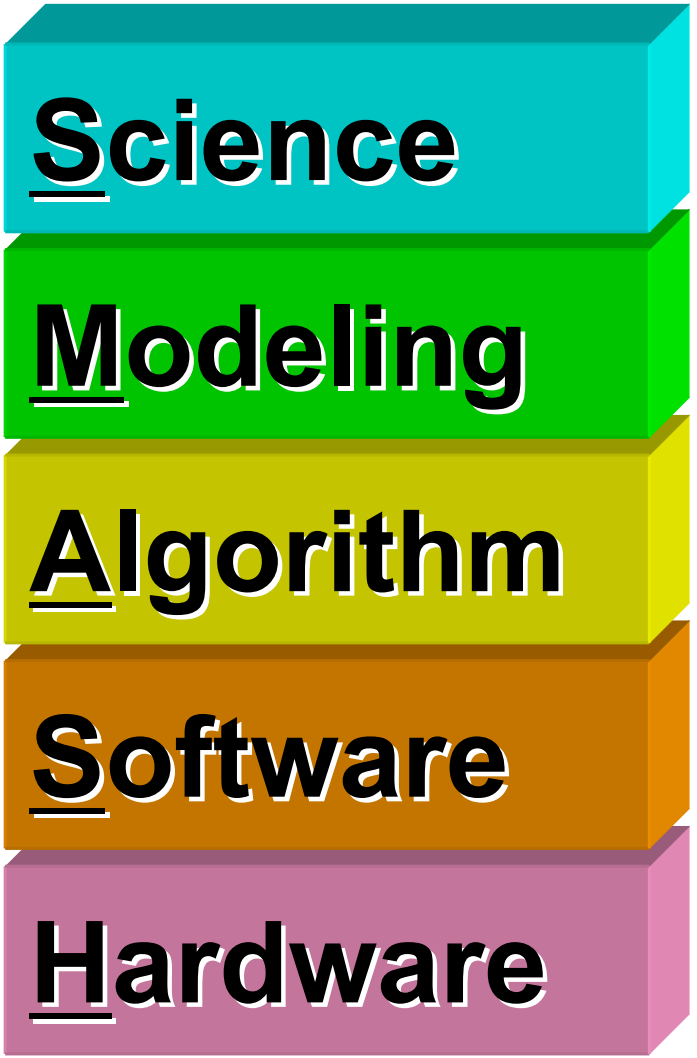




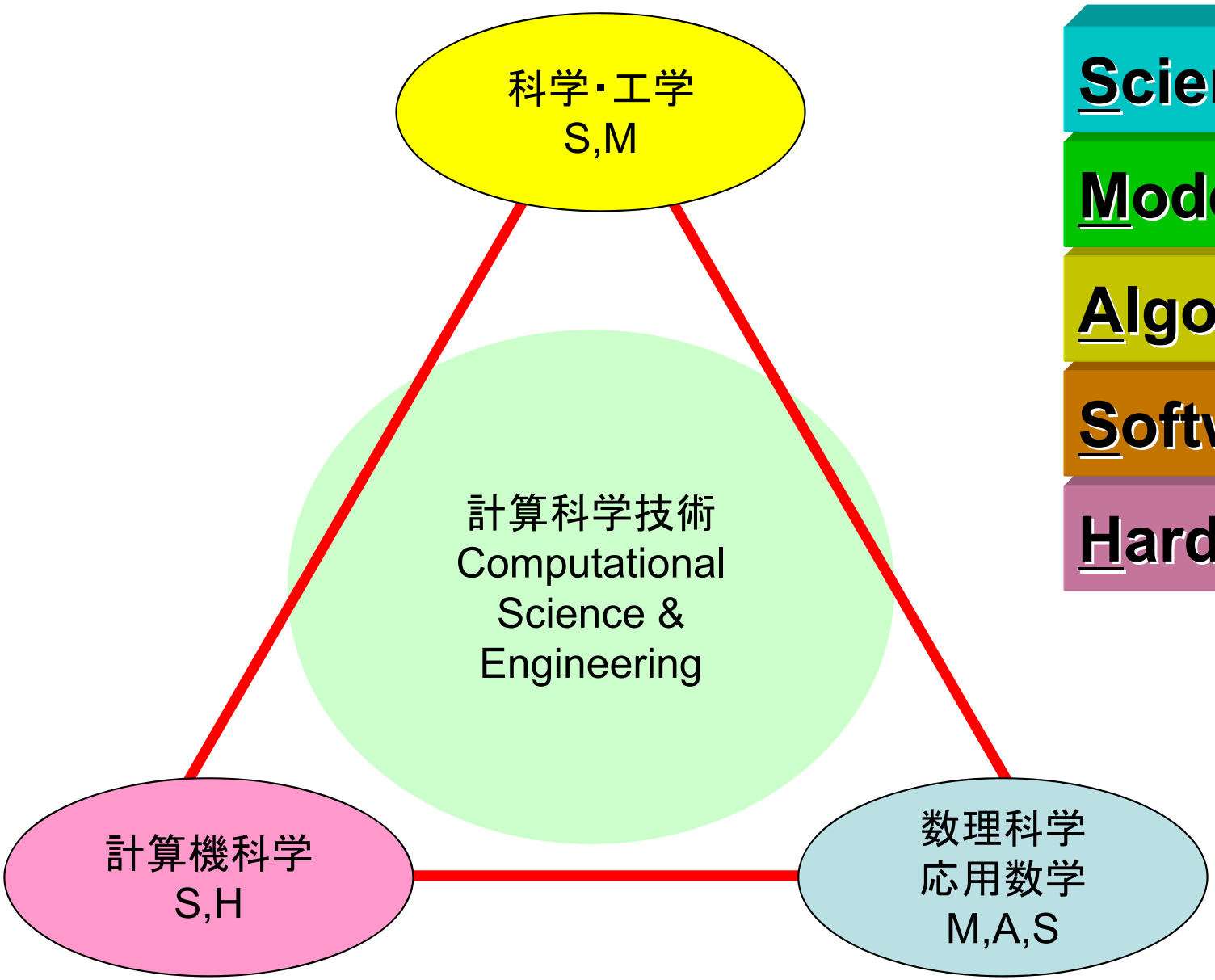
- スーパーコンピュータ(スパコン)とは
- スパコンによる科学技術シミュレーション
- **SMASH**
- プログラミングについて

並列計算機による科学技術シミュレーションに必要なこと

- サイエンスから計算機ハードウェアまで幅広い知識が必要（SMASH）
  - Science　　科学，現象
  - Modeling　　微分方程式，計算モデル
  - Algorithm　　数学アルゴリズム
  - Software　　プログラム
  - Hardware　　計算機本体
- 各階層は独立した専門分野
- 一人で全てやるには限界がある
  - 協力が大切
  - 協力のためには「自分の専門外についてのある程度の知識」がなければならない



様々な分野の協力


科学・工学
S,M
計算科学技術
Computational
Science &
Engineering
計算機科学
S,H
数理科学
応用数学
M,A,S
Science
Modeling
Algorithm
Software
Hardware

計算科学・スーパーコンピューティングを担う人材になるためには

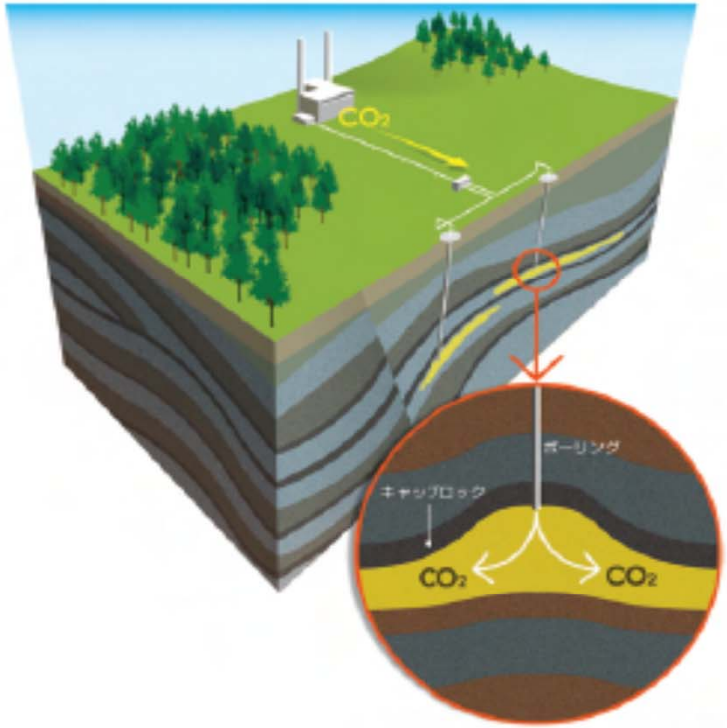
- 色々と勉強しなければならないことは多い：SMASH
  - 何にでも興味を持つことは大切
  - 広く，浅い「何でも屋」が良いわけではない
- T字型人間
  - 一つ，深く掘り下げた専門分野を持っている
  - 関連した分野についても，ある程度の知識・知見を持っている

- 協力が必要：Co-Design
  - T字型人間が集まることによって，集団としての力は更に大きくなる
- どの分野の専門家になるにしても「数学」は必須，「数学」をちゃんと勉強しておこう！・・・と物理屋，計算機屋には常々言っている，諸君は心配ないが

CO_2地下貯留シミュレーション SMASHによる協力の実例

- CO_2を地下（1km以深）に貯留：温暖化ガス削減に寄与
- 地球シミュレータ2
- 大成建設，海洋研究開発機構，NEC，東大情報基盤センター，Lawrence Berkeley国立研究所による共同研究・協力
- 2010年度地盤工学会地盤環境賞受賞

- 現在は東大-大成建設で共同研究を継続中



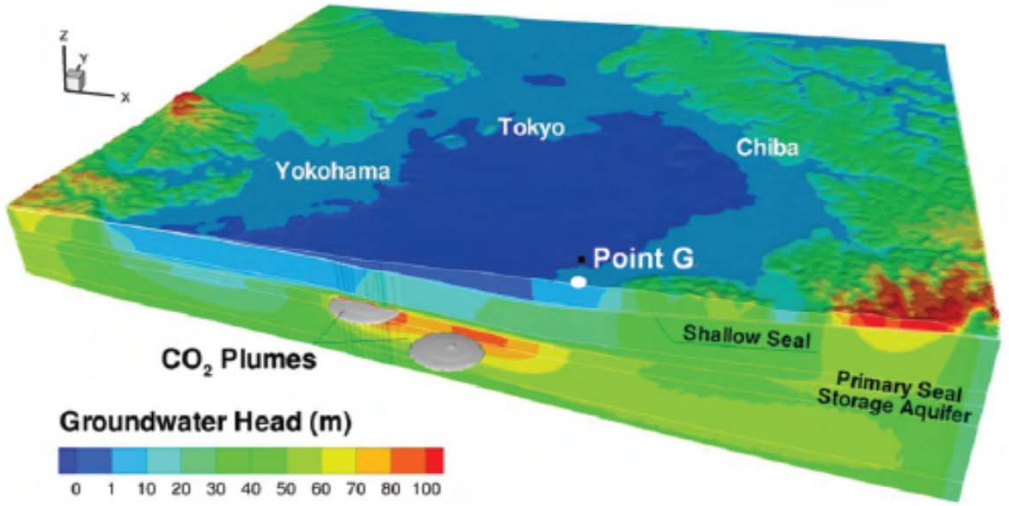
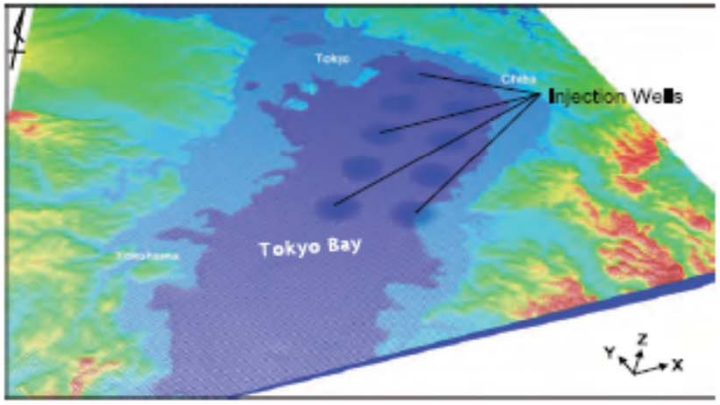
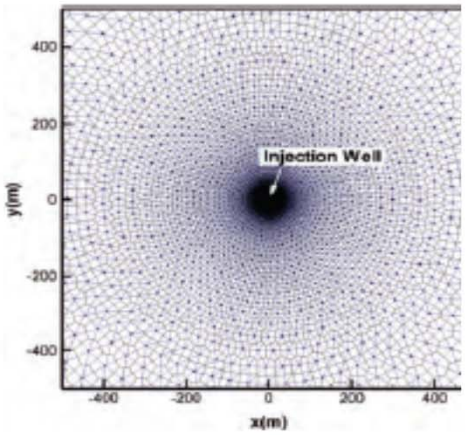
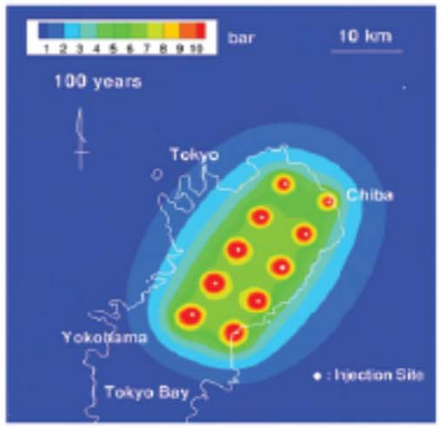
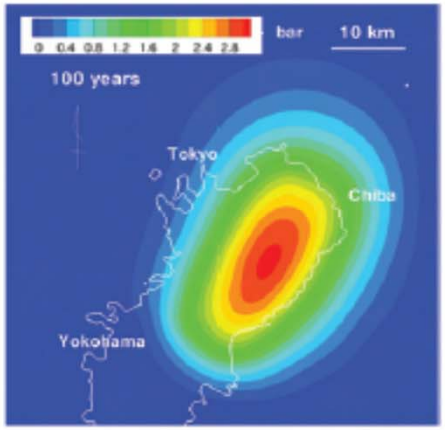


図 -4　CO_2 圧入後の地下水圧（全水頭換算）の分布（100 年後）









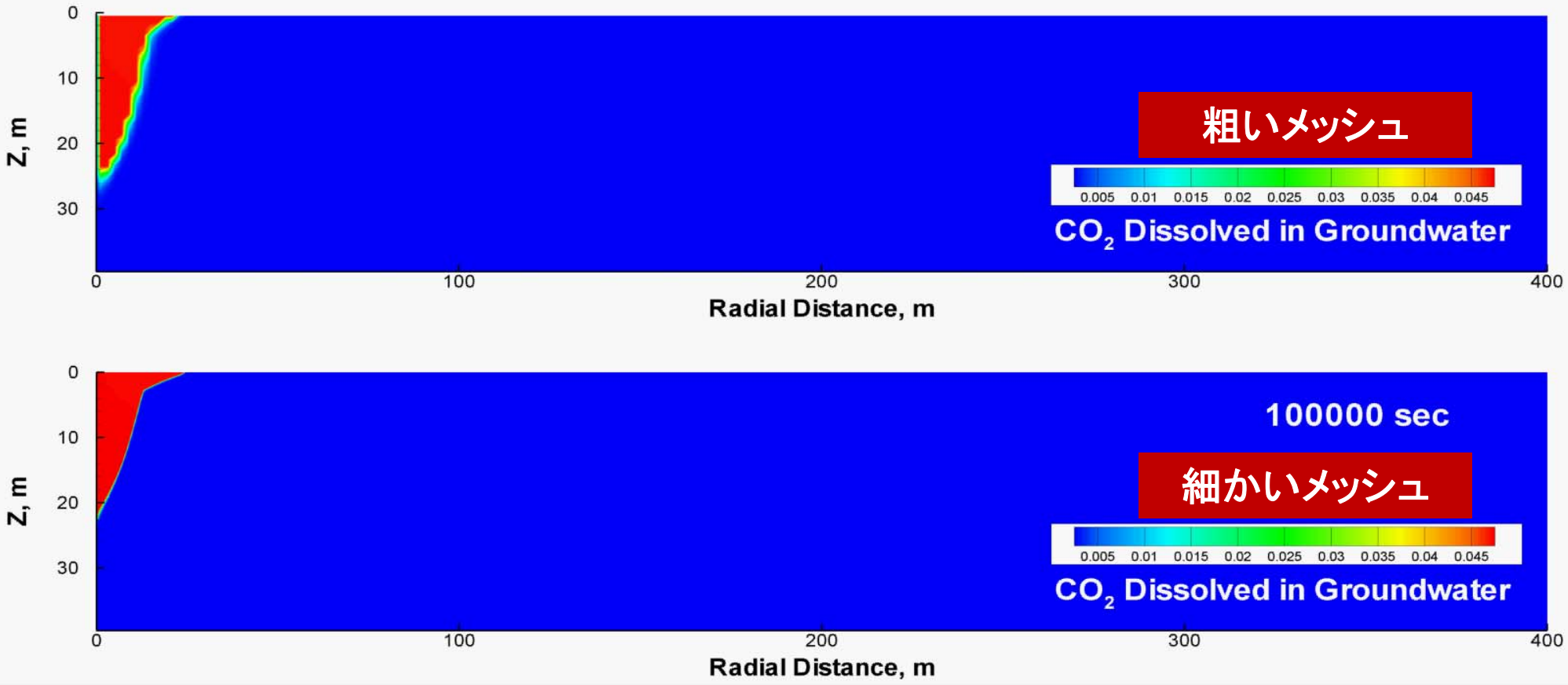
(a) 深部遮蔽層下面　　(b) 浅部遮蔽層下面

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）

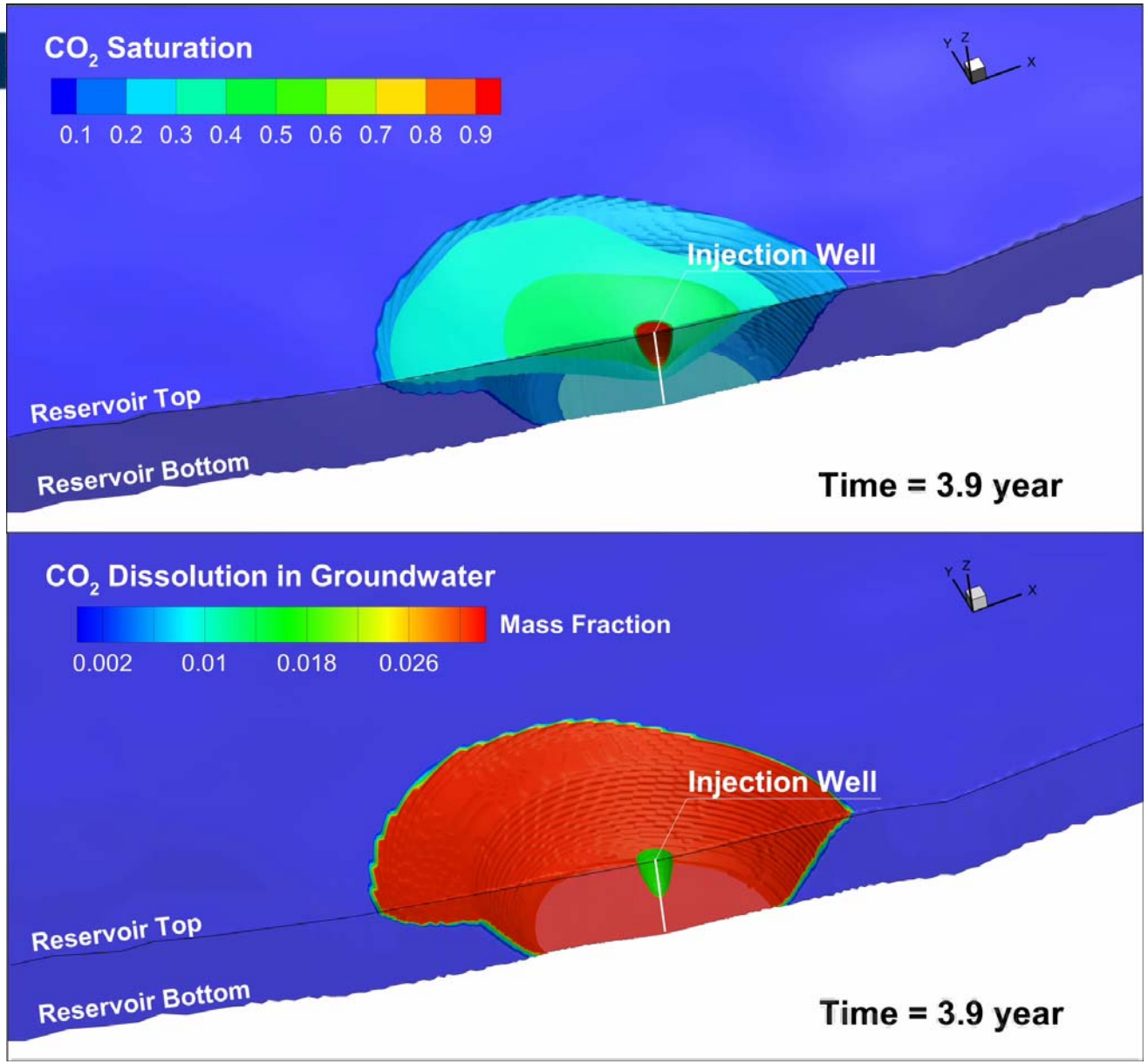
〔画像提供：山本肇博士（大成建設）〕

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



〔画像提供：山本肇博士（大成建設）〕



Density convections for 1,000 years:

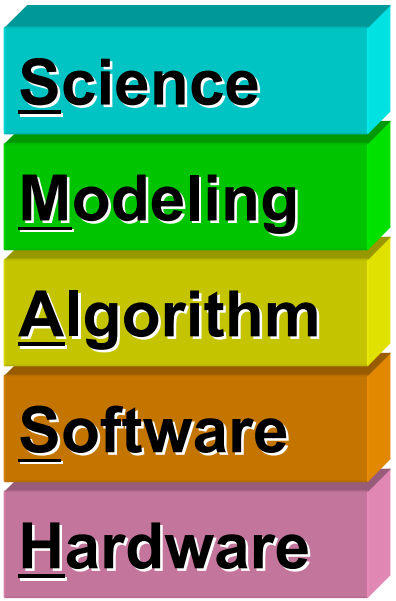
Flow Model

Only the far side of the vertical cross section passing through the injection well is depicted.

[Dr. Hajime Yamamoto, Taisei]

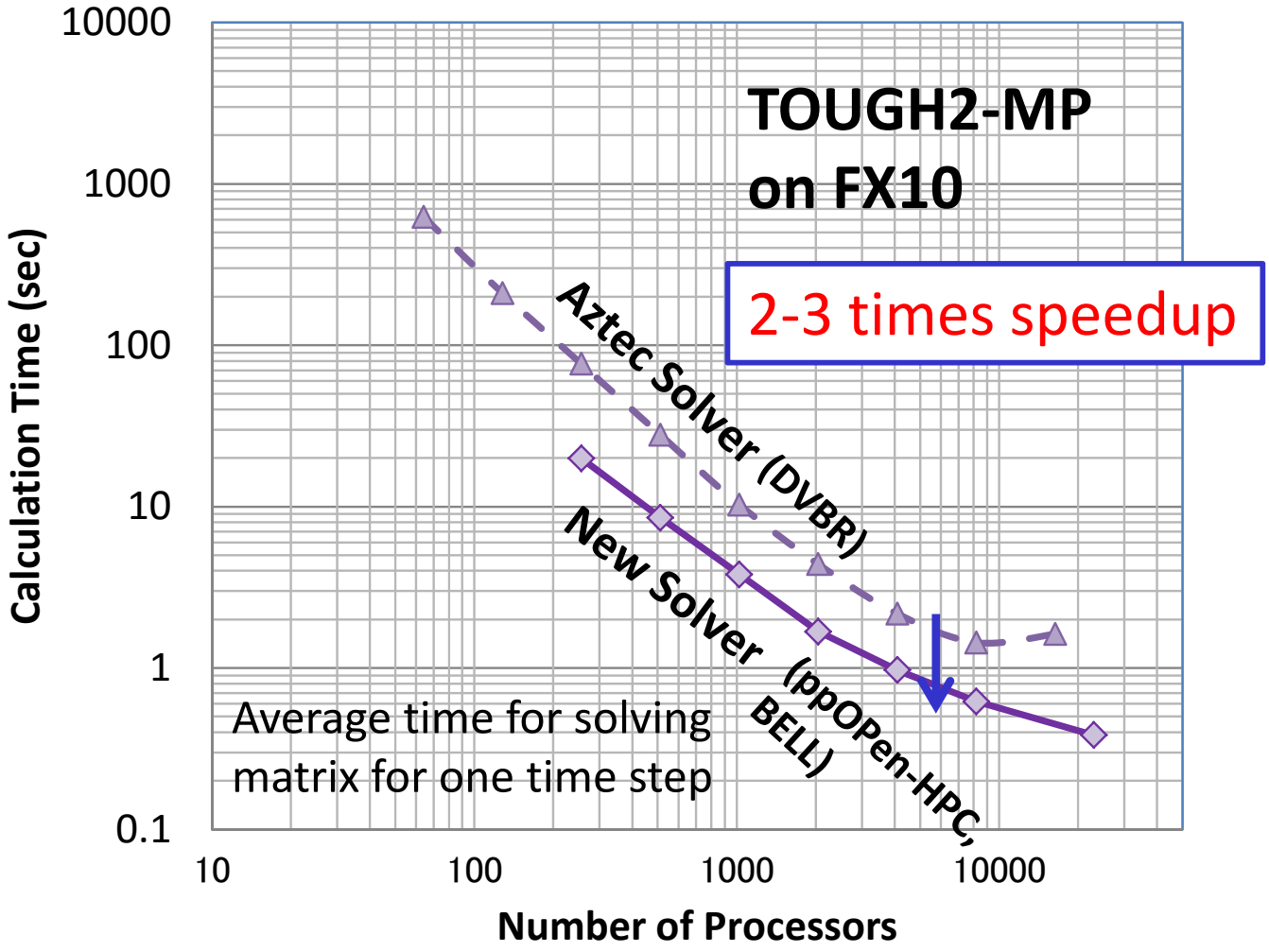
- The meter-scale fingers gradually developed to larger ones in the field-scale model
- Huge number of time steps (> 10^5) were required to complete the 1,000-yrs simulation
- Onset time (10-20 yrs) is comparable to theoretical (linear stability analysis, 15.5yrs)

- Science（科学・現象）
  - 地下深部における超臨界状態のCO_2の挙動（大成建設）
- Modeling（微分方程式）
  - 三次元多相流れ（液体・気体）方程式＋三次元物質移動方程式
- Modeling（計算モデル）
  - 有限体積法によるプログラムTOUGH2（Lawrence Berkeley National Laboratory）
    - 大規模な連立一次方程式（10^7元以上）を解いている部分が全体の90%以上
- Algorithm（数学アルゴリズム）
  - 高速な連立一次方程式求解アルゴリズム（東大）
- Software（プログラム）
- Hardware（計算機本体）
  - 地球シミュレータ2（海洋研究開発機構，NEC）
  - TOUGH2の改良・高速化（Lawrence Berkeley，東大，NEC）

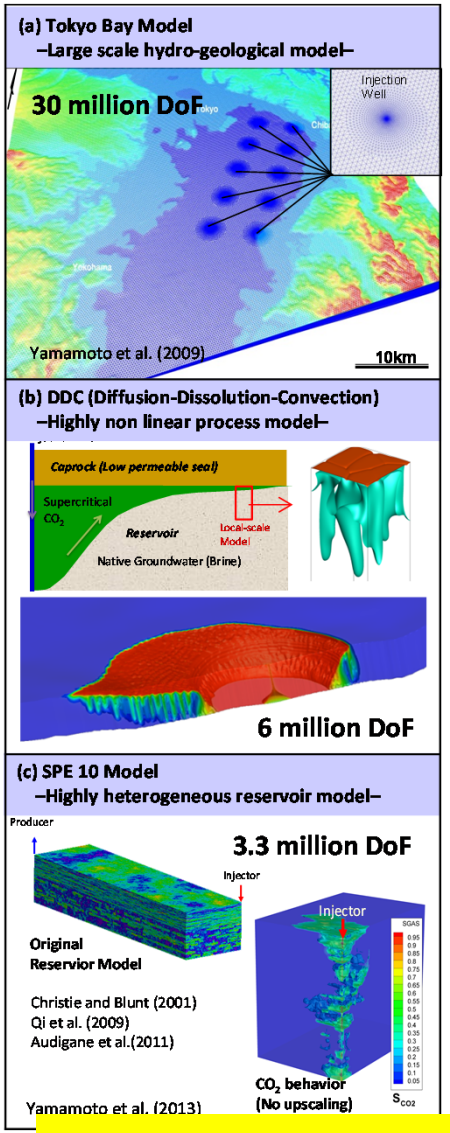


Simulation of Geologic CO_2 Storage 最近の結果

30 million DoF (10 million grids × 3 DoF/grid node)



[Dr. Hajime Yamamoto, Taisei]



※

Fujitsu FX10（Oakleaf-FX），30M DOF: 2x-3x improvement

3D Multiphase Flow (Liquid/Gas) + 3D Mass Transfer

- スーパーコンピュータ(スパコン)とは
- スパコンによる科学技術シミュレーション
- SMASH
- プログラミングについて

プログラミングに関する私見(1/2)

- スーパーコンピューティングにはプログラミングに習熟していること(ただプログラムが書けるという程度ではない)が必須
  - 日本の科学者，計算機科学者：概して数学に弱い(欧米は違う)
  - 日本の数理科学者：概してプログラミングに弱い(欧米は違う)
- 本来であれば，本講義はプログラミングと合わせて教えるのが理想であるが・・・
- 数値計算の環境・ツール
  - 数値計算ソフトウェア：MATLAB, Scilab,（Mathematica）
  - プログラミング言語：Fortran，C，C++，Java
- できれば，本講義においてもC，Fortranでプログラムを作ってほしいのだが，なかなか困難
  - Scilab，MATLABでも許す

プログラミングに関する私見(2/2)

- ただ，本日の講義を聴いて少しでも「スパコン」，「スーパーコンピューティング」に興味を持った人，将来取り組んでみたい人はMATLAB, Scilabにのめり込むことなく，プログラミングをしっかり勉強してほしい(3年になってからでも良いが)
  - http://www.cspp.cc.u-tokyo.ac.jp/ohshima/201509c/
- プログラミング言語
  - スクリプト言語(用途限定)，汎用言語
  - インタープリタ型(一行毎機械語に翻訳)，コンパイラ型(まとめて翻訳)
  - MATLAB，Scilab：インタープリタ型のため遅い，並列化もできない(並列MATLAB開発の要望もあるが)，また，例えば新しい並列反復法のアルゴリズムの開発には全く使えない
  - MATLAB, Scilabの文法はC，Fortranとそれほど異なっていない

計算環境について(1/3)

- OS
  - Windows, Mac OS X, Unix/Linux
  - スパコンのOSはUnix/Linux
- プログラミング言語
  - C, C++, Java, Fortran, Pascal, Python
- 数値計算ソフトウェア
  - Mathematica, Matlab, Scilab

ー長ー短あります（こちらでは特に指定しません）
何で計算してもOK！（気合いがあれば電卓でも可能）

- ECCS（教育用計算機システム）
- 個人のPC

計算環境について(2/3)

- Windows
  - Cygwin （Windows上でのUnixライクな環境）を入手すれば，C, C++, Java, Fortran（g77, g95, gfortran），　Pascal, Pythonなどが使えます （フリー）
  - Intelコンパイラ （C,C++,Fortran）（有料）
- Mac OS X
  - Unix/Linuxに近い環境
  - C, C++, Java, Fortran（g77, g95, gfortran）, Pascal, Pythonなどが使えます（GNU, フリー）
  - Intelコンパイラ （C,C++,Fortran）（有料）
- Unix/Linux
  - C, C++, Java, Fortran（g77, g95, gfortran）, Pascal, Pythonなどが使えます（GNU, フリー）
  - Intelコンパイラ（C,C++,Fortran）（有料，フリー版もあり）

計算環境について(3/3)

- 数値計算ソフトウェア
  - Mathematica（数式処理）
  - MATLAB（有料，高い）
  - Scilab（フリー）
    （やや不安定だが学習には充分，Matlabクローン）

  - それぞれWindows, Mac OS X, Unix/Linux版があります

ECCS：教育用計算機システム

- http://www.ecc.u-tokyo.ac.jp/
- http://www.ecc.u-tokyo.ac.jp/guide/tebiki/

- Mac OS，Windows

- MATLAB（同時利用制限69名），Mathematica
- Fortran，C，C++，Java

お勧め

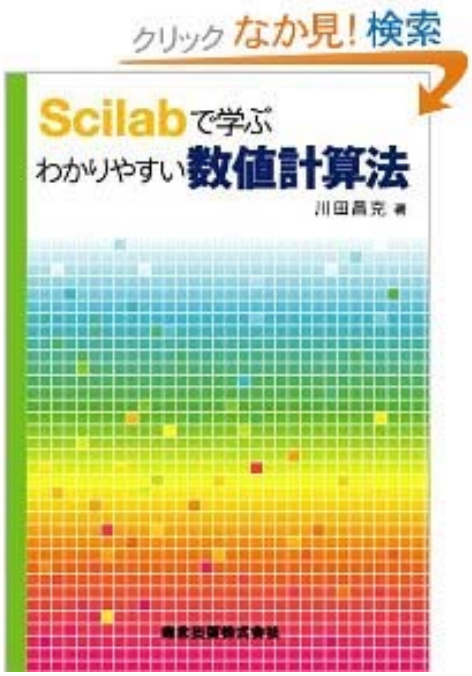
- ECCS
  - MATLAB（利用者数制限あり）
  - Scilab（自分でインストール必要）
  - Fortran，C，C++，Java 他

- 自分のPC
  - Scilab
  - Fortran，C，C++，Java 他

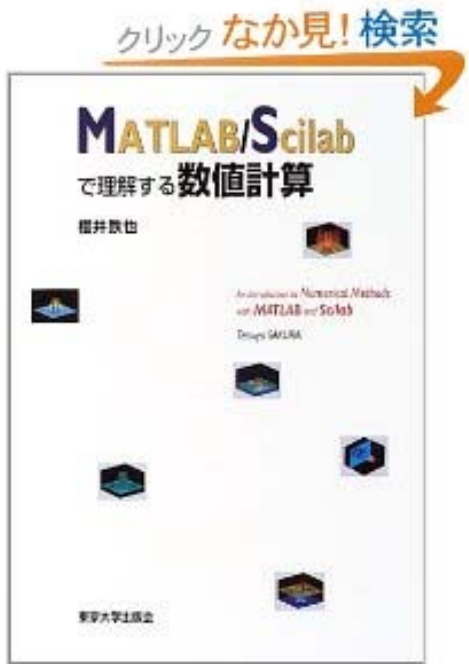
参考文献（自習用）

「Cによる数値計算法」
鈴木・飯田・石塚 陽
（オーム社）

「C言語による数値計算入門―解法・アルゴリズム・プログラム」皆本
（サイエンス社）



「Scilabで学ぶわかりやすい数値計算法」川田（森北）
中島はこれを使用
2008年発行だが内容がやや古い

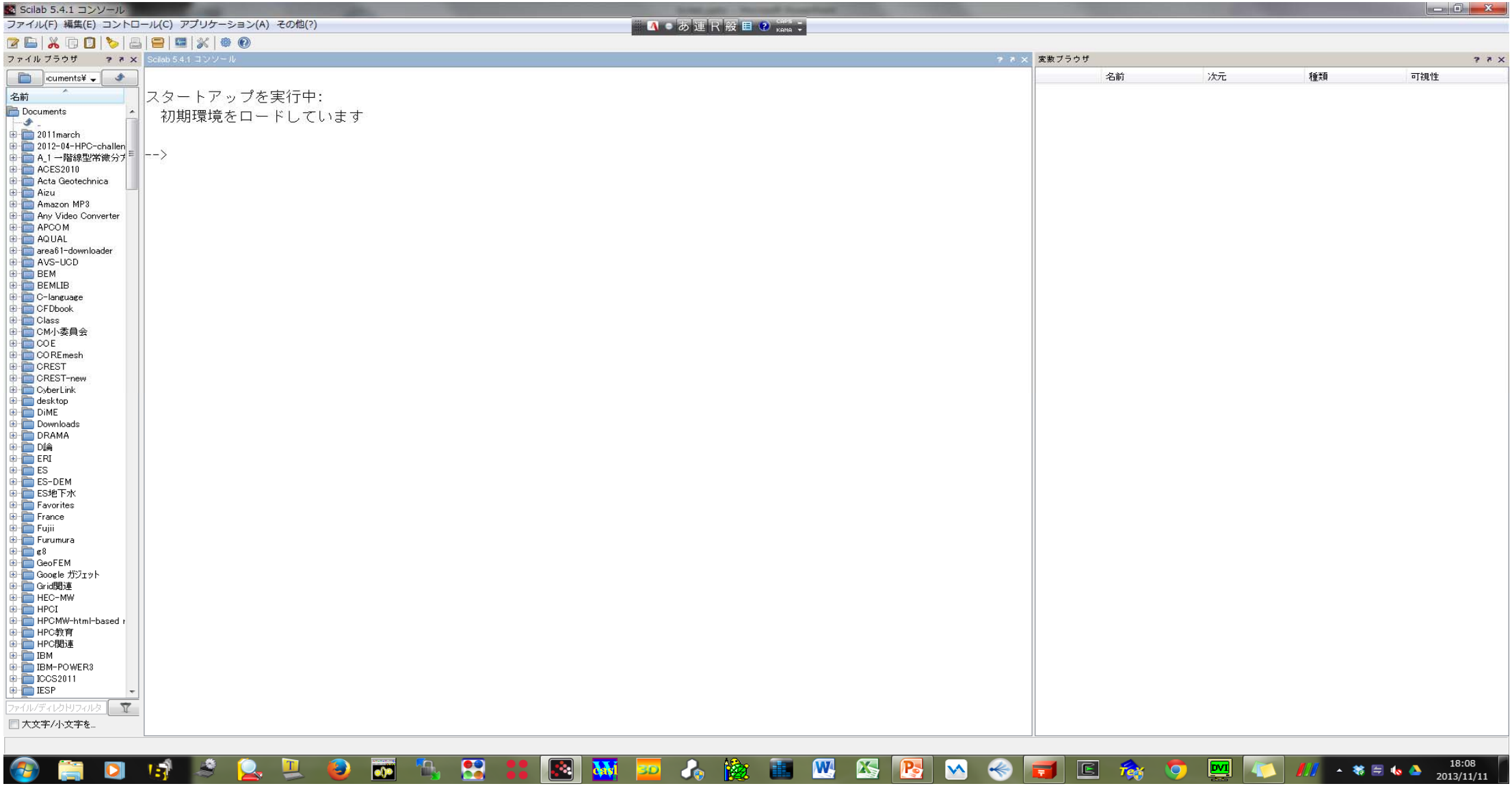


「MATLAB/Scilabで理解する数値計算」櫻井（東大出版）
2003年発行なので更に古いかも

Scilab

- http://www.scilab.org/
  - 現在 Ver.5.5.2
  - Versionが変わると結構文法が変わっているので注意
- 電卓的な使用法
- プログラム作成・実施

Scilab : 初期画面


Scilab 5.4.1 コンソール
ファイル(F) 編集(E) コントロール(C) アプリケーション(A) その他(?)
ファイル ブラウザ
Scilab 5.4.1 コンソール
変数ブラウザ
名前
次元
種類
可視性
スタートアップを実行中:
初期環境をロードしています
-->
Documents
2011march
2012-04-HPC-challen
A_1 一階線型常微分方
ACES2010
Acta Geotechnica
Aizu
Amazon MP3
Any Video Converter
APCOM
AQUAL
area61-downloader
AVS-UCD
BEM
BEMLIB
C-language
CFDbook
Class
CM小委員会
COE
COREmesh
CREST
CREST-new
CyberLink
desktop
DiME
Downloads
DRAMA
D論
ERI
ES
ES-DEM
ES地下水
Favorites
France
Fujii
Furumura
g8
GeoFEM
Google ガジェット
Grid関連
HEC-MW
HPCI
HPCMW-html-based
HPC教育
HPC関連
IBM
IBM-POWER3
ICCS2011
IESP
ファイル/ディレクトリフィルタ
大文字/小文字を...
18:08
2013/11/11

基本的演算

- 電卓としての使用

```
1+2
3*4
(1+2*%i)*(1-2*%i)   %i:虚数単位
                    (MATLABではi)
%pi (MATLAB : pi)
```

- 変数への保存(「型」は無い)

```
a=1;
b=2;
c=a+b;
c
```

- 行列ベクトル演算

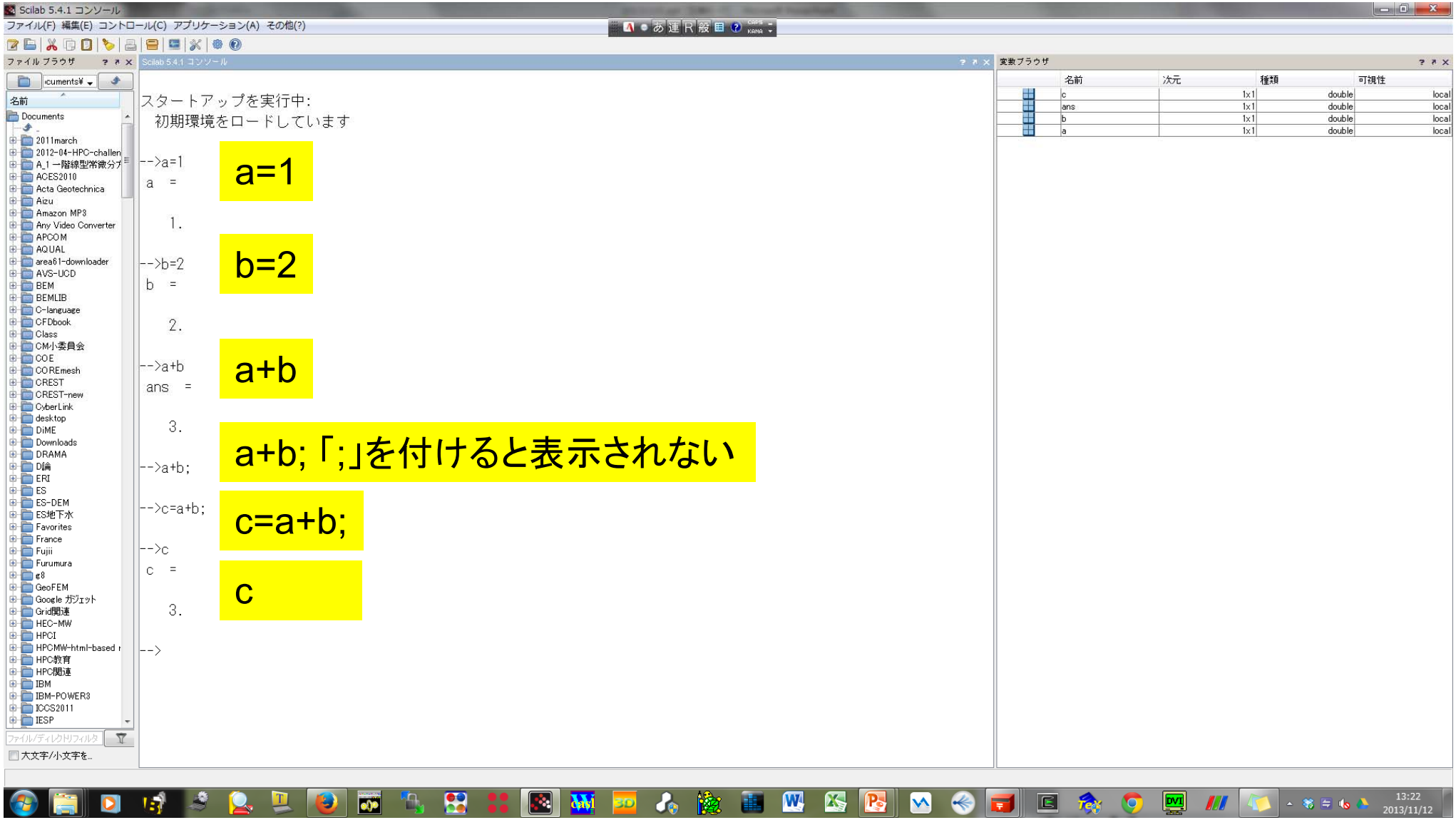
$(1,2,3) \Rightarrow [1\ 2\ 3]$

$(1,2,3)^T \Rightarrow [1;2;3]$

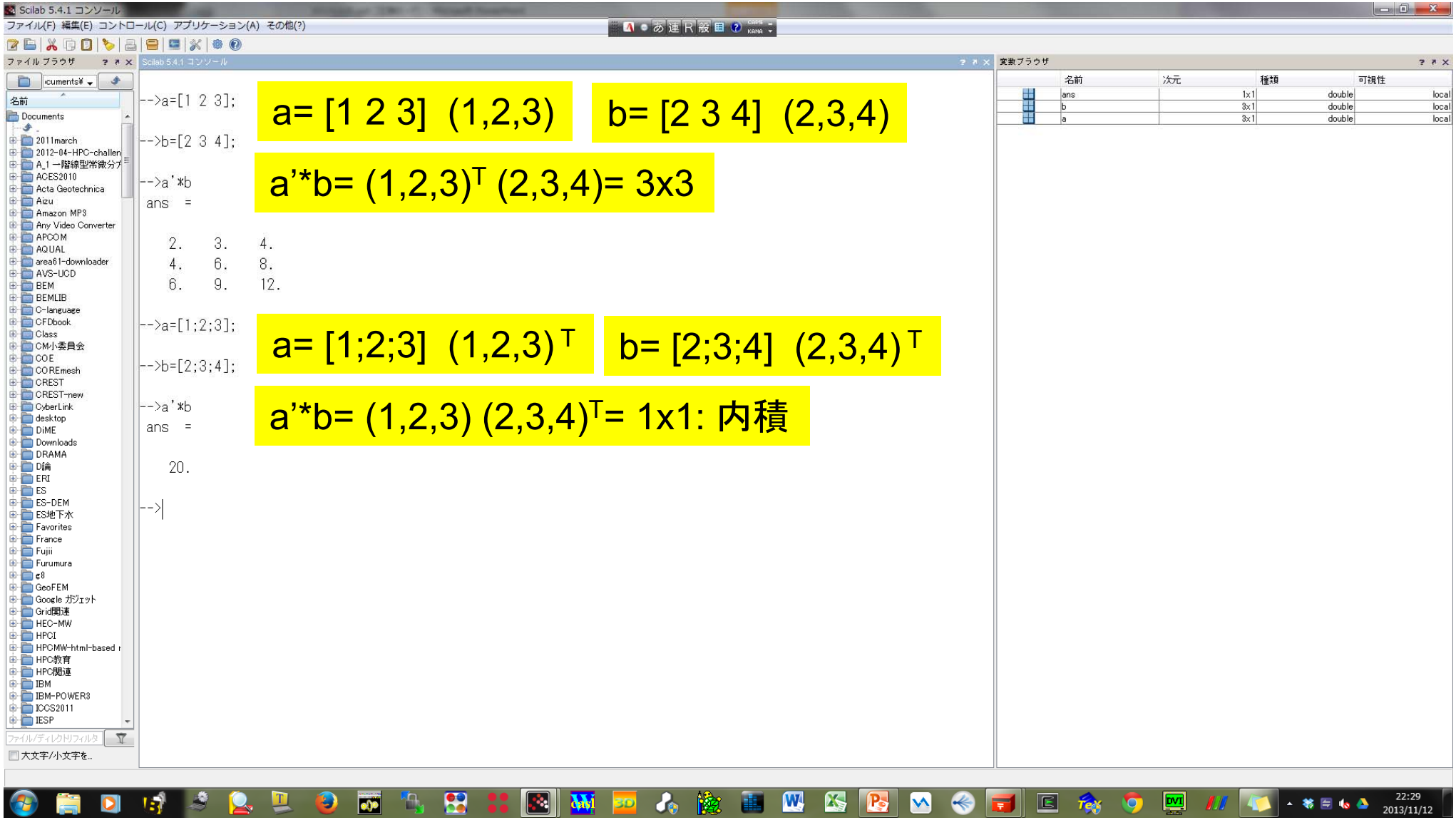
$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \Rightarrow [1\ 2;3\ 4]$

「’ 」	共役転置
A(2,3)	A_{23}
B(2)	B_2
zeros(2,3)	全成分が「0」の2x3行列
ones(4,5)	全成分が「1」の4x5行列
eye(4,4)	4x4単位行列
rand(5,2)	5x2乱数行列
diag([1 2 3])	対角成分が[1,2,3]の3x3対角行列

Scilab : a=1, b=2, a+b ?



Scilab：ベクトル演算



プログラム:制御構造

- 条件分岐(if文)(MATLABはthen不要)

```
if  条件式  then
  命令
 elseif 条件式 then
  命令
...
 elseif 条件式 then
  命令
 else
  命令
end
```

- 条件式(a,b:スカラー)

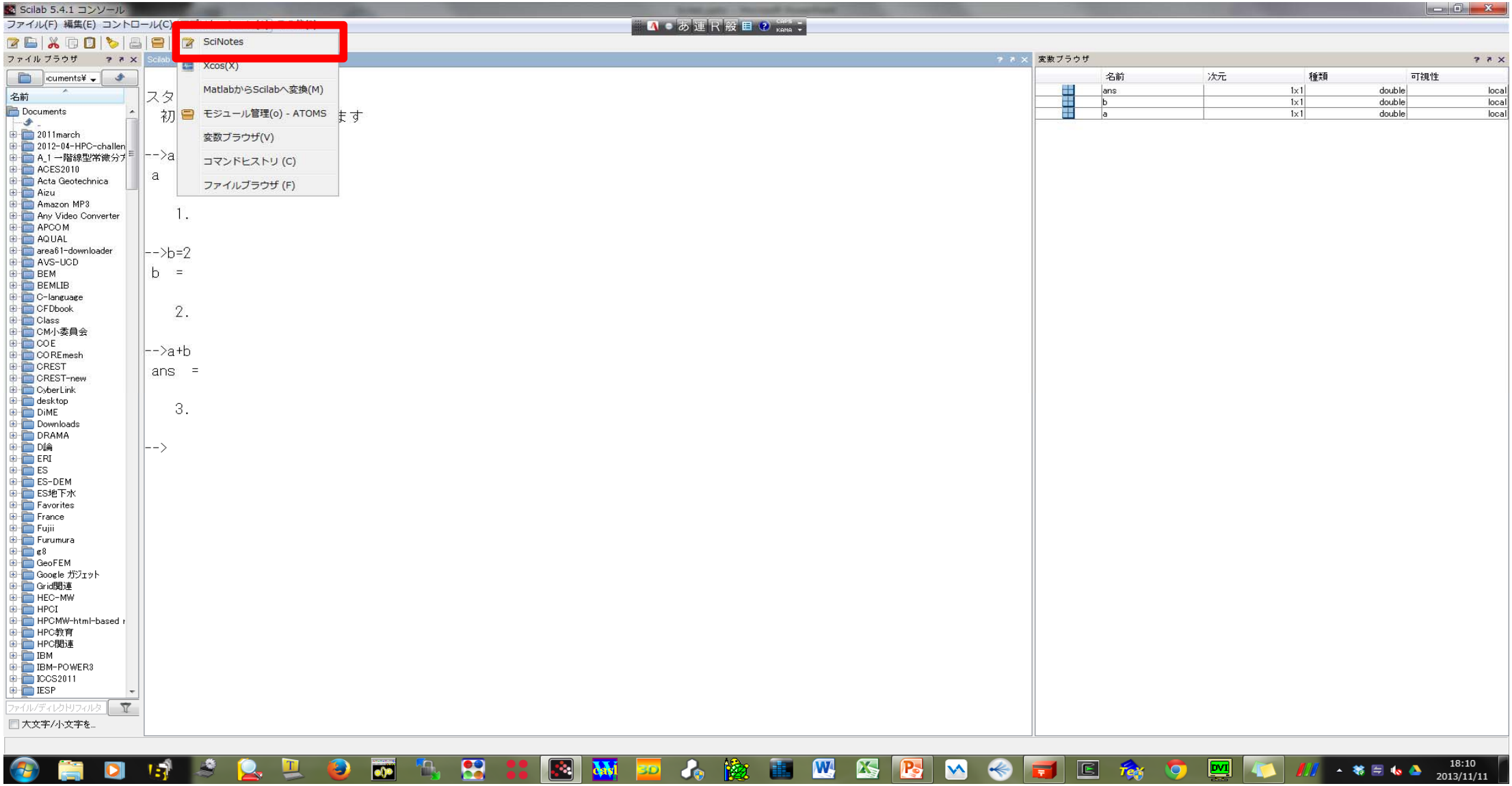
```
a<b a>b a<=b a>=b
a==b a~=b

if ((a>b+1) & (C~=0)) then ...(AND)
if ((a>b+1) | (C~=0)) then ...(OR)
```

- 繰り返し(ループ)

```
for i= 1:10
   b(i)= a(i);
end

while (r>1.0)
  r= r + 0.1;
end

continue        次のループ（i）へ移動
break           ループから抜ける
```

Scilab：プログラムの編集

SciNotes (SciPadではない)



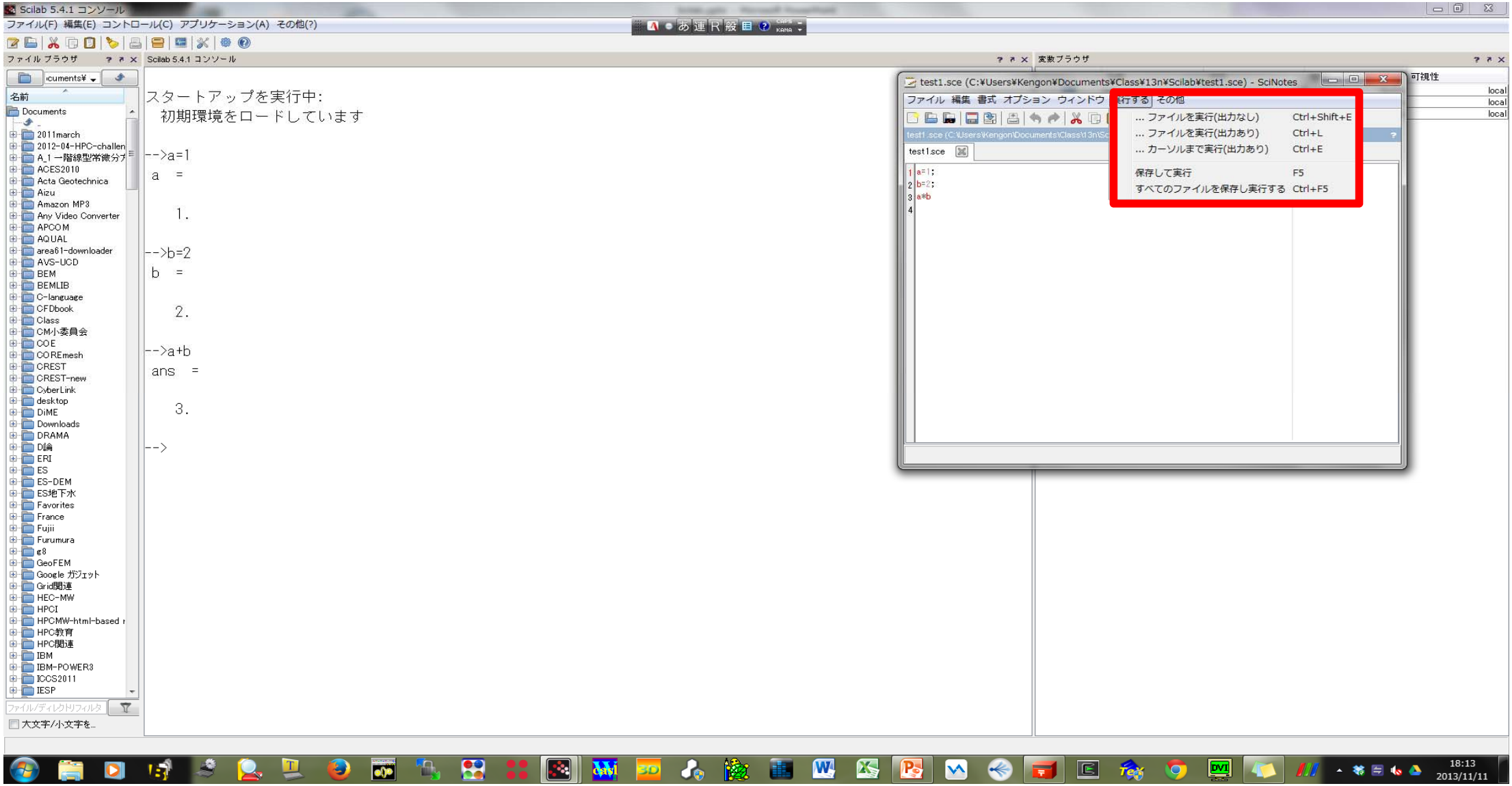
Scilab：プログラム例（1）

test1.sce （拡張子は「sce」）

```
a=1;
b=2;
a*b
```

Scilab：プログラムの編集＋実行

test1.sci



Scilab：プログラム例（2）

testfact.sce

```
// fact.sciを呼び出す例

exec('C:¥Users¥xx¥Documents¥yy¥Scilab¥fact.sci'); //Scilabでは必須

fact(5)
fact(10)
```

//:コメント，該当行のここから後ろは無視

exec:「スクリプト（プログラム）」の実行，フルパスで入力
（以前は「getf」を使っていた）

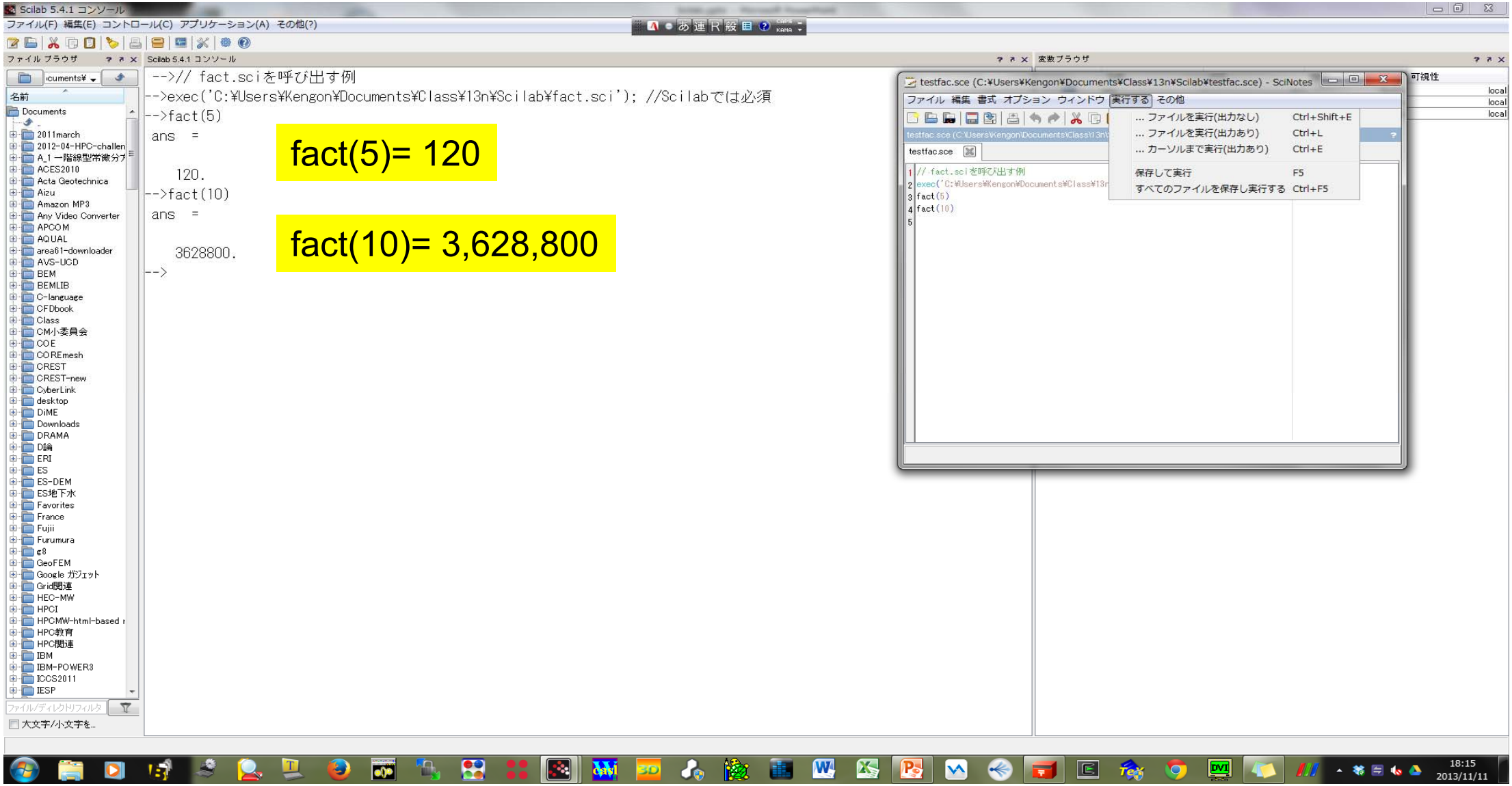
Scilab：関数の例（2）

fact.sci（関数の拡張子「sci」（sceでもok））

```
//  関数定義サンプル：　階乗

function [f] = fact(n)
    if (n==1) then
        f = 1; // 関数実行中に結果が表示されないよう;で抑制
    else
        f = n * fact(n-1);
    end
endfunction
```

Scilab：プログラムの編集＋実行

testfac.sce



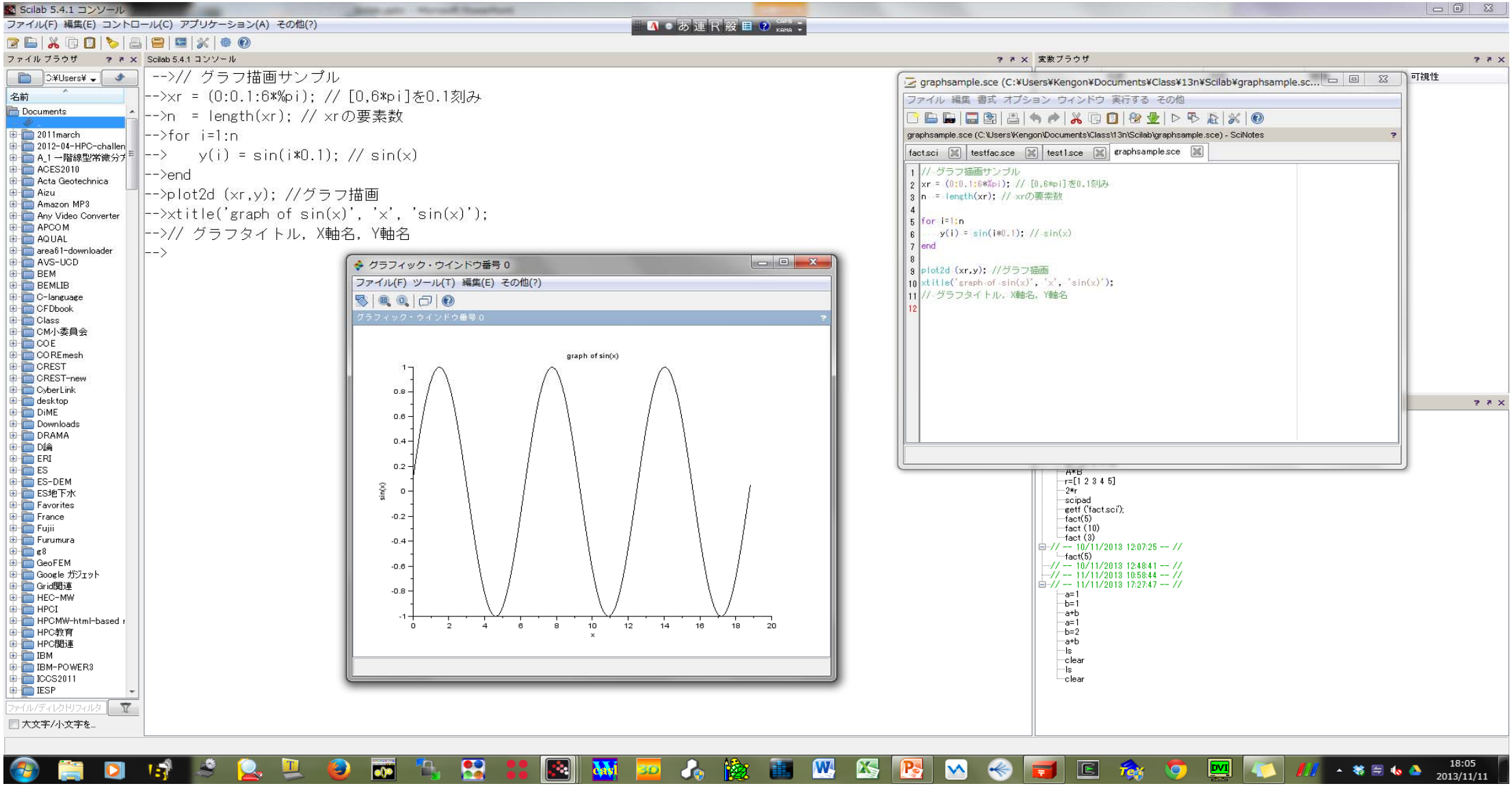
Scilab:プログラム例(3)

graphsample.sce

```
// グラフ描画サンプル
xr = (0:0.1:6*%pi); // [0,6*pi]を0.1刻み
n  = length(xr); // xrの要素数

for i=1:n
    y(i) = sin(i*0.1); // sin(x)
end

plot2d (xr,y); //グラフ描画

xtitle('graph of sin(x)', 'x', 'sin(x)');
// グラフタイトル, X軸名, Y軸名
```

Scilab：プログラムの編集＋実行

graphsample.sce



Scilab:プログラム例(4)(1/2)

timer.sce

```
// timer 関数で実行時間を測定
// 行列ベクトル積の時間計測（2通りの実行方法）

n=100; // ←ここを色々と変えてみる

A=rand(n,n); // n×nの乱数行列
b=rand(n,1); // 長さnの乱数ベクトル

//例1：内部ルーチン使用
timer(); // 時間測定開始
c=A*b;
t1=timer(); // 前回のtimer()からの実行時間測定
```

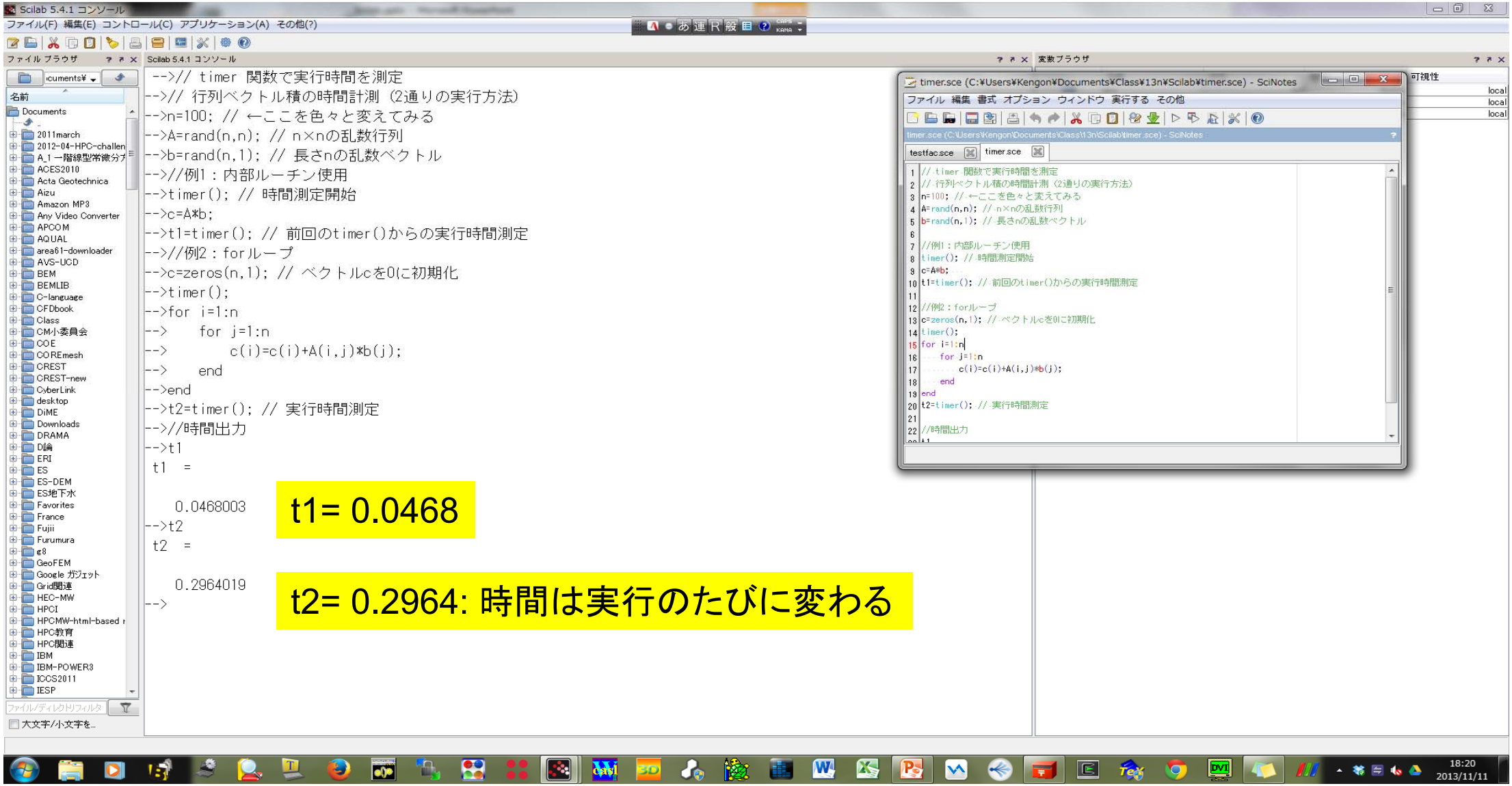
Scilab:プログラム例(4)(2/2)

timer.sce

```
//例2：forループ
c=zeros(n,1); // ベクトルcを0に初期化
timer();
for  i=1:n
    for  j=1:n
        c(i)=c(i)+A(i,j)*b(j);
    end
end
t2=timer(); // 実行時間測定

//時間出力
t1
t2
```

Scilab：プログラムの編集＋実行

timer.sce，内部ルーチンの方がだいぶ速いが，めげてはいけない



http://nkl.cc.u-tokyo.ac.jp/16n/