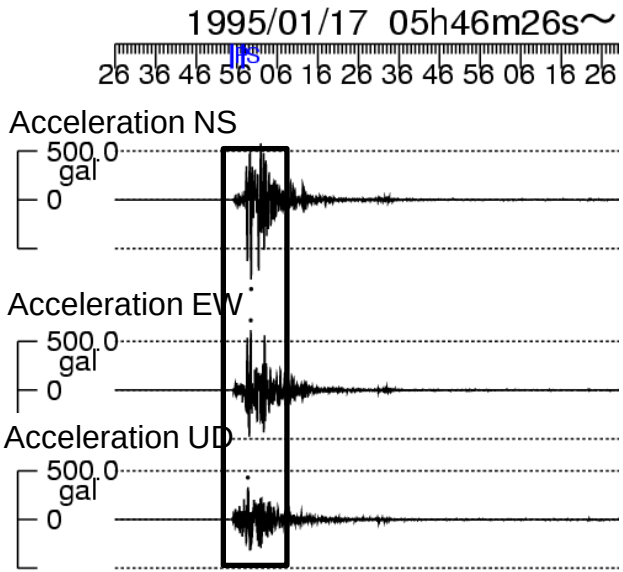
2015年8月17日(月)
RIKEN AICS HPC Summer School 2015

地震シミュレーションとHPC：京における都市地震シミュレータの開発に着目して

理化学研究所　計算科学研究機構
総合防災・減災研究ユニット
藤田　航平

イントロダクション

- 兵庫県南部地震(1995年)相当の地震が東京で起こったら…



Visualized by CYBERNET SYSTEMS CO., LTD

目次

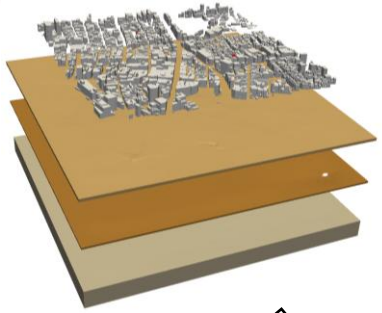
1. 地震災害現象の概要
2. 地震シミュレーションとHPC
3. 京における都市地震シミュレーションの大規模化・高速化
4. 大規模・高速な都市地震シミュレーションによってできるようになること
5. 今後に向けて：他のシミュレーションとの組み合わせによる，統合地震シミュレーション

地震災害のプロセス

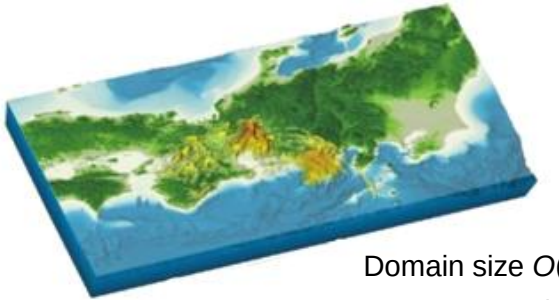
断層破壊
・断層の大きさ・滑り方で地震特性が変化

地盤増幅
・地表に近い地盤の特性に応じて波が増幅

Domain size *O*(1km)



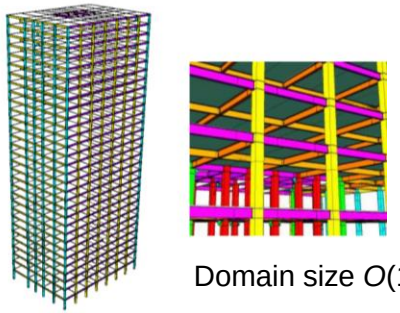
人的・経済的応答
・人間を取り巻く構造物の被害に応じて社会が応答



Domain size *O*(100km)

Furumura et al.

地殻中の波動伝播
・地殻の物性や層構造に応じて波が伝わる

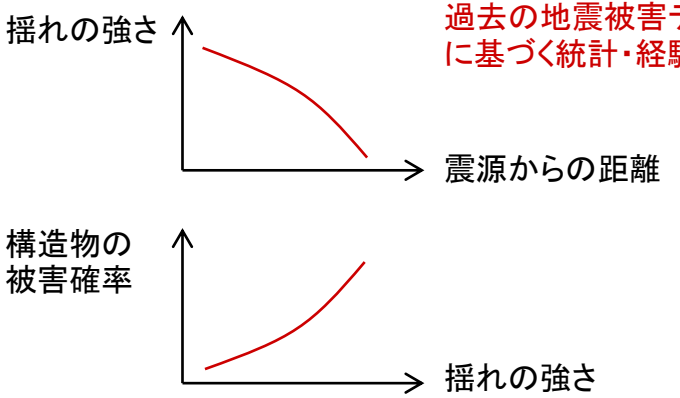


Domain size *O*(10m)

構造物応答
・揺れに応じて被害が生じる

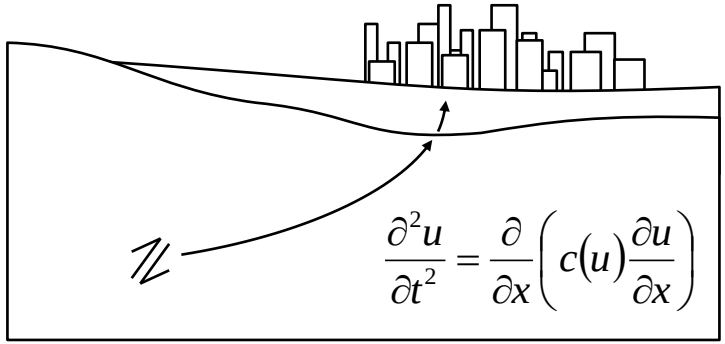
地震災害への対策

- 人的・経済的被害の主な要因は構造物の被害
- 想定した地震に対する構造物被害を推定する技術が地震災害対策のベースとなる
  - 地震災害プロセスは物理現象の組み合わせだが，物理を直接解くには巨大な計算コストがかかる
  - そこで実務では統計・経験式ベースの方法が被害推定に使われてきた
  - スパコンの計算能力を活用した地震シミュレーションにより，推定の信頼性向上が期待できる



過去の地震被害データ
に基づく統計・経験式

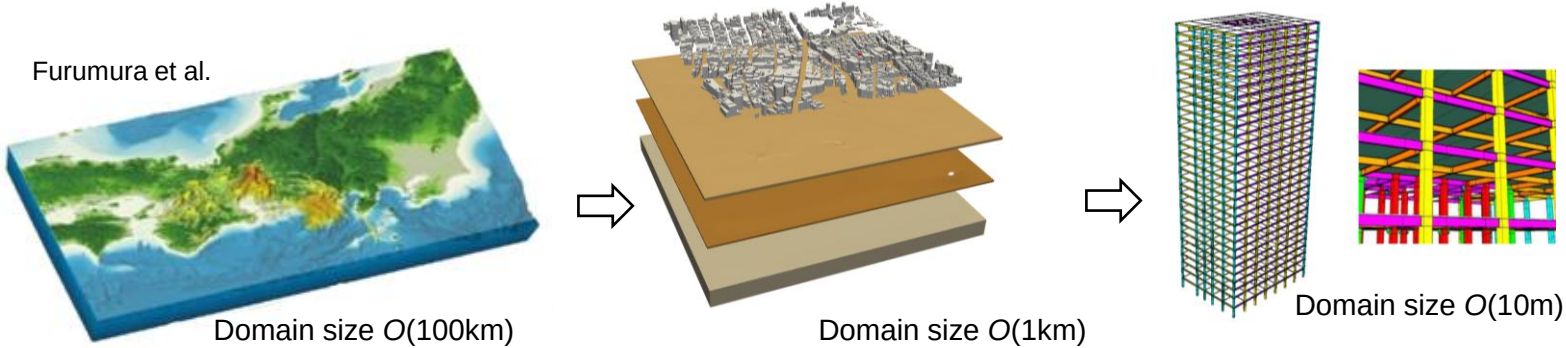
統計・経験式ベースの被害推定



物理シミュレーションベースの被害推定

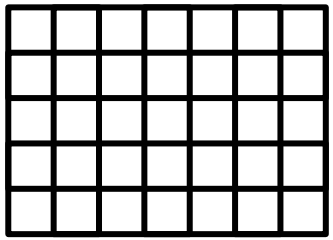
地震シミュレーションとHPC (1)

- 地震シミュレーション：巨大な数値解析問題になるため，HPC(高性能計算)と共に発展してきた
  - 地殻中の波動伝播解析がHPCの主な研究対象
  - 構造物の応答解析は構造設計目的で発展
- 地盤増幅は構造物の被害に大きな影響を与えるにもかかわらず，HPCではあまり余り取り組まれてこなかった
  - 地盤は柔らかく非線形化するので，地殻解析よりもさらに計算コストがかかる
  - ただし，これを解決できれば，地震シミュレーション全体の信頼性向上が期待できる
  - まずは地殻解析で培われてきたHPC技術をレビュー



地震シミュレーションとHPC (2)

- 何しろ計算コストが膨大
  - コストの少ない方法は…
- 有限差分法
  - データ配置が**規則的**で性能が出しやすいため，HPC黎明期から使われ続けている
  - 一方で地表面形状の取り扱いが難しい



有限差分法 (構造格子)

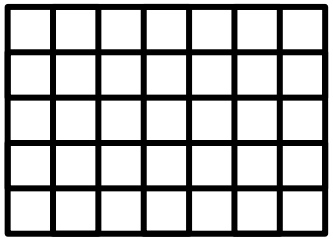
プログラム上では
do i=1,nx
do j=1,ny
a(i,j)=b(i,j)+b(i+1,j)+….
enddo
enddo
など

コラム1

- データ移動のレイテンシとスループット
  - レイテンシ：データ移動を要求してから，その結果が返送されるまでの遅延時間
  - スループット：単位時間当たりのデータ移動量
- **規則的**なデータ配置に対してはデータ移動のレイテンシが隠しやすく，スループットをあげやすい

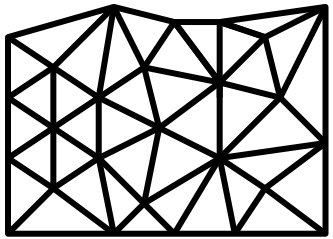
地震シミュレーションとHPC (3)

- 形状の問題を解決するため，有限要素法(非構造格子＋陰的時間積分＋低次要素)が使われるように
  - 表面形状の取り扱いに優れるが，メモリアクセスが複雑かつ全計算ノード間の通信あり
  - データアクセスコスト・通信コスト共に高く，高い性能を出すには工夫が必要



有限差分法 (構造格子)

プログラム上では
do i=1,nx
do j=1,ny
a(i,j)=b(i,j)+b(i+1,j)+….
enddo
enddo
など



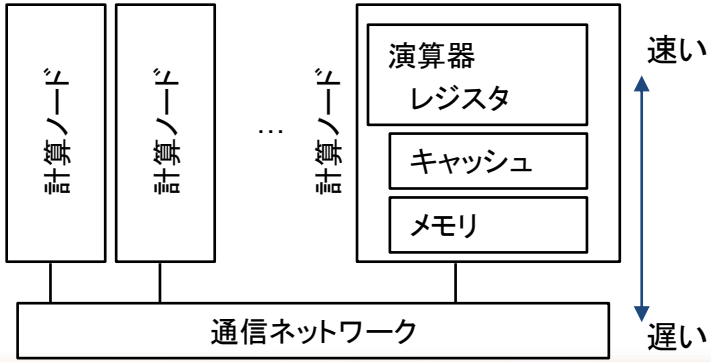
有限要素法 (非構造格子)

プログラム上では
do i=1,n
do j=ptr(i)+1,ptr(i+1)
a(i)=a(i)+b(index(j))
enddo
enddo
など

コラム2

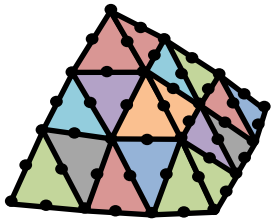
- スパコン(並列計算機)の階層構造
  - レジスタ→キャッシュ→メインメモリ→通信ネットワーク
  - レイテンシ・スループット性能はこの順で遅くなっていく

➥ メモリアクセス・通信は遅い…

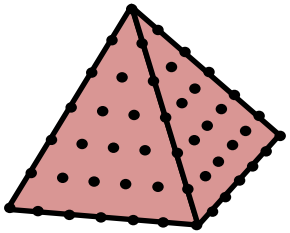


地震シミュレーションとHPC (4)

- スペクトル要素法(非構造格子＋陽的時間積分＋高次要素)
  - 要素の次数(精度)を上げ，さらに，陽的な時間積分を使えるよう工夫した方法
  - B/Fが小さく，全計算ノード間の通信が必要ない
  - 現在の並列計算機に向いた方法



有限要素法(低次要素)
・要素一個当たりの計算コストが小さい = B/F大
・精度を出すには多数の要素が必要



スペクトル要素法(高次要素)
・要素一個当たりの計算コストが大きい = B/F小
・少数の要素で高い精度

コラム3

- B/F (Byte per FLOP)
  - 単位計算量(FLOP)あたりのデータ移動量(Byte)
  - 現在の計算機は計算よりもデータ移動の方がコストが大きいので，B/Fが小さい方法ほど高い計算性能を出しやすい

コラム4

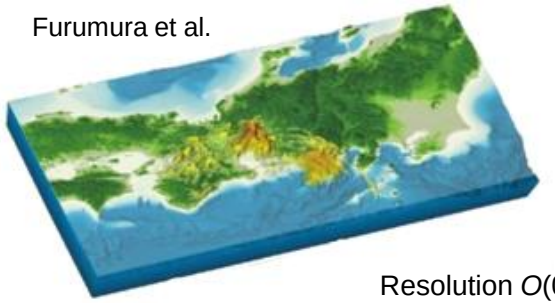
- 陰的時間積分
  - マトリクス方程式を解く
  - 全計算ノード間＋隣り合う計算ノード間での通信が必要
- 陽的時間積分
  - マトリクス方程式を解かない
  - 隣り合う計算ノード間でのみ通信

地震シミュレーション：次のステップ

- スペクトル要素法は地殻解析に適用できるが，地盤の解析には不向き
  - 地殻に比べて地盤は形状が複雑で，かつ，地震時に軟化する(非線形化)
  - これらを計算するには必要な計算精度以上に要素サイズを小さくする必要があり，無駄が多い
- 性能は出しにくいものの，地盤解析には有限要素法が向いている
  - メモリアクセス・通信をケアして，京の性能を引き出すことができれば，都市規模の計算も可能となるはず

➥ 次のステップ：有限要素法(非構造格子＋陰的時間積分＋低次要素)による地盤解析

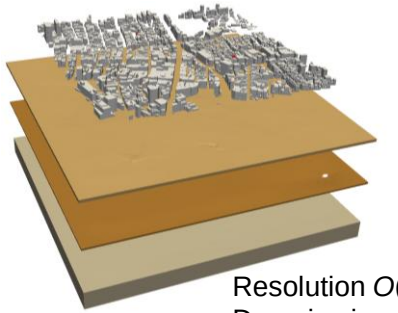
Furumura et al.



Resolution *O*(0.1km),
Domain size *O*(100km)

地殻解析

⇔

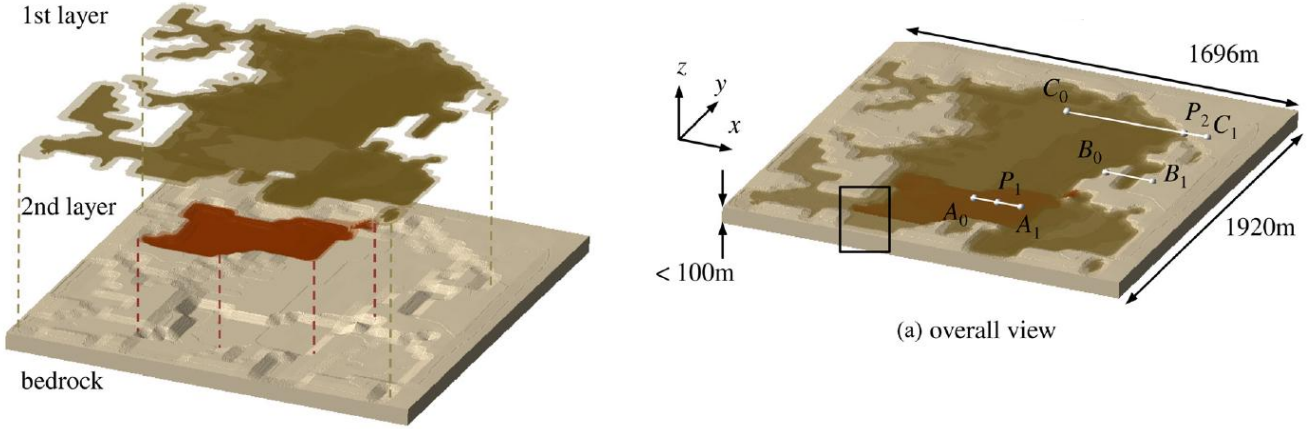


Resolution *O*(1m),
Domain size *O*(1km)

地盤解析

地盤解析の現状

- 陰的時間積分・低次要素を使った有限要素法による地盤解析[1]
  - 空間分解能4m, 時間分解能2.5Hz
- 計算負荷が大きい
  - 構造物の固有周期である10^1Hzまでは計算できない→構造物解析の入力として使うことができない
  - 多数ケースの解析を実行できない → 解析結果の信頼性確認が難しい



[1] Ichimura, T. et al., *Journal of Pressure Vessel Technology,* 2014.

ターゲット問題

- 陰解法・非構造格子有限要素法により連続体中の波動方程式を離散化
  - 時間分解能15Hzの都市規模問題は，以下のようになる

未知ベクトル(100億自由度)

$$\left(\frac{4}{dt^2}\mathbf{M}+\frac{2}{dt}\mathbf{C}^n+\mathbf{K}^n\right)\delta\mathbf{u}^n=\mathbf{F}^n-\mathbf{Q}^{n-1}+\mathbf{C}^n\mathbf{v}^{n-1}+\mathbf{M}\left(\mathbf{a}^{n-1}+\frac{4}{dt}\mathbf{v}^{n-1}\right)$$

疎行列 (非線形化により毎タイムステップ変化)

外力ベクトル

タイムステップ毎に相対誤差10^{-8}で求解
- 各タイムステップで物性を線形化して解析
- 時間刻みdt=10^{-3}秒で数万～数十万ステップ

CG法

- CG法(共役勾配法)
  - 反復法によるマトリクス方程式の求解方法(ソルバー)の一つ
  - メモリ使用量が少なく，大規模解析でよく使われる
- CG法における前処理
  - 求解対象の*A*を変換することで，等価だが収束までの反復回数が少ない問題に変換する方法
  - 実際には行列*A*に近い行列*M*の逆行列を反復の内部で使用することで計算する
- 前処理の効果
  - *M*=*A*の場合前処理は非常に効果的 (反復1回で求解できる)が，前処理自体のコスト大
  - *M*=*I*(単位行列)の場合は前処理のコストは小さいが，前処理の効果は無くなる
  - 前処理コストに対して反復回数を効率良く削減できる方法がCG法の高速化に効果的

CG法のアルゴリズム　(b=Axの求解)

$\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$
$\mathbf{z}_0 := \mathbf{M}^{-1}\mathbf{r}_0$
$\mathbf{p}_0 := \mathbf{z}_0$
$k := 0$
repeat
$\alpha_k := \frac{\mathbf{r}_k^{\mathrm{T}}\mathbf{z}_k}{\mathbf{p}_k^{\mathrm{T}}\mathbf{A}\mathbf{p}_k}$
$\mathbf{x}_{k+1} := \mathbf{x}_k + \alpha_k\mathbf{p}_k$
$\mathbf{r}_{k+1} := \mathbf{r}_k - \alpha_k\mathbf{A}\mathbf{p}_k$
if $\mathbf{r}_{k+1}$ is sufficiently small then exit loop end if
$\mathbf{z}_{k+1} := \mathbf{M}^{-1}\mathbf{r}_{k+1}$ 前処理
$\beta_k := \frac{\mathbf{z}_{k+1}^{\mathrm{T}}\mathbf{r}_{k+1}}{\mathbf{z}_k^{\mathrm{T}}\mathbf{r}_k}$
$\mathbf{p}_{k+1} := \mathbf{z}_{k+1} + \beta_k\mathbf{p}_k$
$k := k + 1$
end repeat
The result is $\mathbf{x}_{k+1}$

http://en.wikipedia.org/wiki/Conjugate_gradient_method

CG法(続き)

- Inexact preconditioned CG法
  - CG法における前処理方法の一種
  - $z=M^{-1}r$を計算する代わりに，前処理方程式$r=Az$をCG法により解くことで前処理を行う
  - 前処理方程式は荒く解けばよいので工夫の余地あり
    - 使えそうな方法：マルチグリッド法，精度混合演算など…

CG法のアルゴリズム　(b=Axの求解)

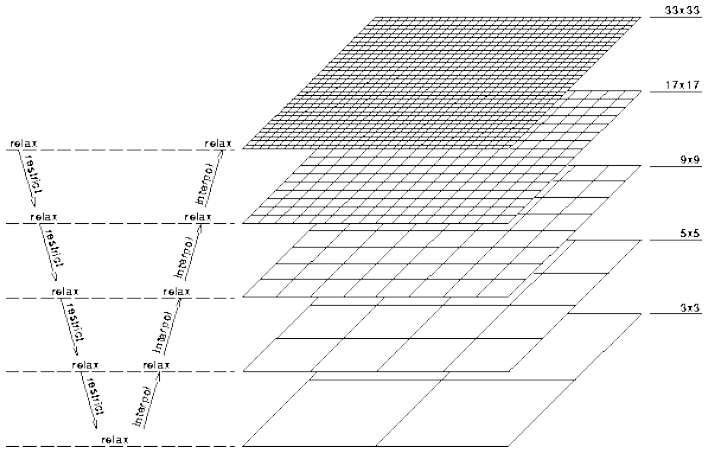
$\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$
$\mathbf{z}_0 := \mathbf{M}^{-1}\mathbf{r}_0$
$\mathbf{p}_0 := \mathbf{z}_0$
$k := 0$
repeat
$\alpha_k := \frac{\mathbf{r}_k^{\mathrm{T}}\mathbf{z}_k}{\mathbf{p}_k^{\mathrm{T}}\mathbf{A}\mathbf{p}_k}$
$\mathbf{x}_{k+1} := \mathbf{x}_k + \alpha_k\mathbf{p}_k$
$\mathbf{r}_{k+1} := \mathbf{r}_k - \alpha_k\mathbf{A}\mathbf{p}_k$
if $\mathbf{r}_{k+1}$ is sufficiently small then exit loop end if
$\mathbf{z}_{k+1} := \mathbf{M}^{-1}\mathbf{r}_{k+1}$　前処理
$\beta_k := \frac{\mathbf{z}_{k+1}^{\mathrm{T}}\mathbf{r}_{k+1}}{\mathbf{z}_k^{\mathrm{T}}\mathbf{r}_k}$
$\mathbf{p}_{k+1} := \mathbf{z}_{k+1} + \beta_k\mathbf{p}_k$
$k := k + 1$
end repeat
The result is $\mathbf{x}_{k+1}$

http://en.wikipedia.org/wiki/Conjugate_gradient_method

マルチグリッド法/精度混合演算

マルチグリッド法

・計算量・データ移動量を減らす



http://www.mgnet.org/mgnet/tutorials/xwb/mg.html

精度混合演算

- 必要な精度に応じて演算精度を選択
- 低い精度を使うことで，計算・データ移動の高速化が可能

例：Intel Xeon E5-2680 v3 CPUの場合
倍精度計算の理論ピーク性能: 480 GFLOPS
単精度計算の理論ピーク性能: 960 GFLOPS
(※FLOPS: 1秒あたりの浮動小数点計算数)

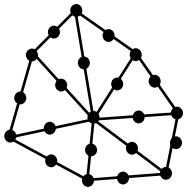
GAMERA

- 非線形地盤震動解析用のソルバー
- HPC手法を組み合わせることで開発
  - multi-**G**rid method
  - **A**daptive conjugate gradient method
  - **M**ulti-precision arithmetic
  - **E**lement-by-element method
  - p**R**edictor with **A**dams-bashworth method

Tsuyoshi Ichimura, Kohei Fujita, Seizo Tanaka, Muneo Hori, Maddegedara Lalith, Yoshihisa Shizawa, and Hiroshi Kobayashi, Physics-based urban earthquake simulation enhanced by 10.7 BlnDOF x 30 K time-step unstructured FE non-linear seismic wave simulation, SC'14, Gordon Bell Finalist.

GAMERAのアルゴリズム

- 前処理方程式を荒く・高速に解くことで，CG法の反復数を削減
  - 前処理にマルチグリッドを使用することで，前処理の計算コストを削減
  - 前処理に単精度を使うことで計算と通信を減らす



CG loop

Solving preconditioning matrix

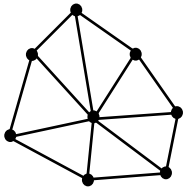
Computation of outer CG loop

Outer loop

Solving preconditioning matrix

Inner coarse loop

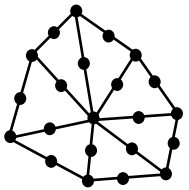
Solve system roughly using CG solver (linear tetrahedral mesh, single precision)



Use as initial solution

Inner fine loop

Solve system roughly using CG solver (second ordered tet. mesh, single precision)

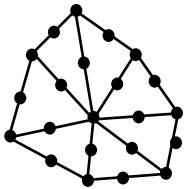


Use for preconditioner of outer loop

Equation to be solved
(second ordered tet. mesh, double precision)

GAMERAのアルゴリズム

- 1ループあたりの通信量(byte比)　Outer : Inner_fine : Inner_coarse = 1 : 1/2 : 1/8
- 1ループあたりの計算量(FLOP比)　Outer : Inner_fine : Inner_coarse = 1 : 1 : 0.18
- Inner_fineとInner_coarseは単精度計算→B/Fが低く，キャッシュ容量が実質的に2倍になる



CG loop

Solving preconditioning matrix

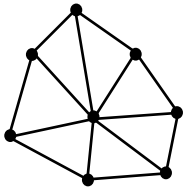
Computation of outer CG loop

Outer loop

Solving preconditioning matrix

Inner coarse loop

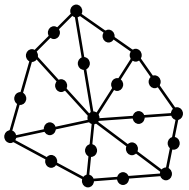
Solve system roughly using CG solver (linear tetrahedral mesh, single precision)



Use as initial solution

Inner fine loop

Solve system roughly using CG solver (second ordered tet. mesh, single precision)



Use for preconditioner of outer loop

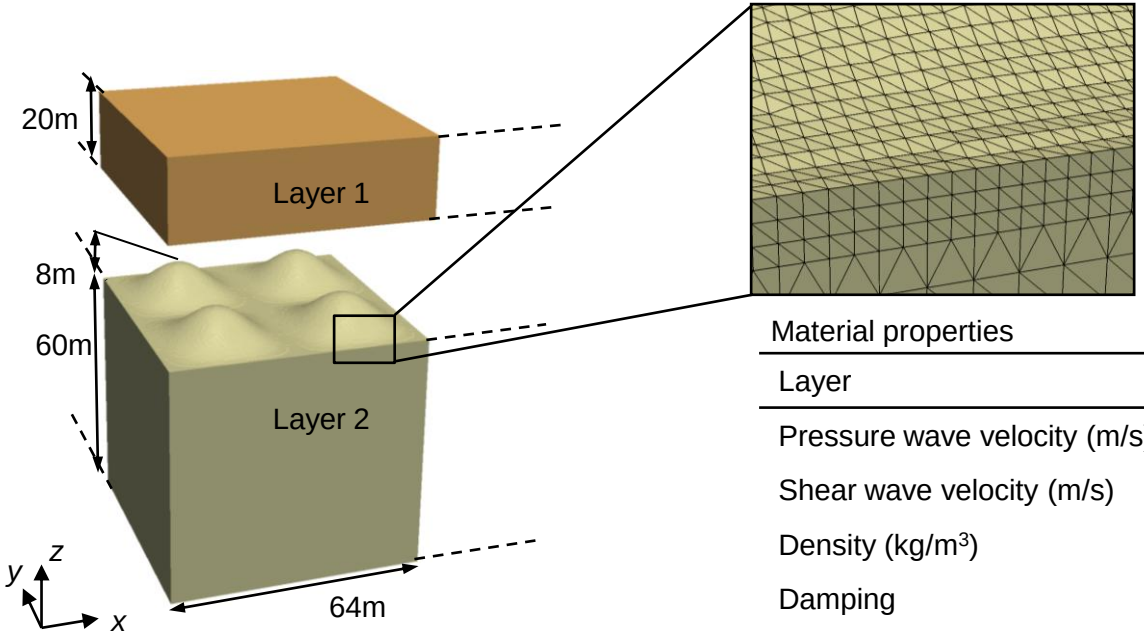
Equation to be solved
(second ordered tet. mesh, double precision)

計算環境

- 京コンピュータ@理化学研究所・計算科学研究機構
  - 82,944 compute nodes, each with single SPARC64 VIIIfx CPU, connected with Tofu interconnect (6-dimensional mesh/torus network)
  - Peak performance of each CPU: (FMA SIMDs with vector length 2) x 2 sets x 8 cores x 2.0GHz = 128GFLOPS
  - Peak performance of system: 128GFLOPS x 82,944 nodes = 10.6PFLOPS

性能測定

- 問題設定
  - *x*, *y*方向に並べることで，周期性のある問題を設定
  - METISで領域分割：実問題と同様のロードバランス特性



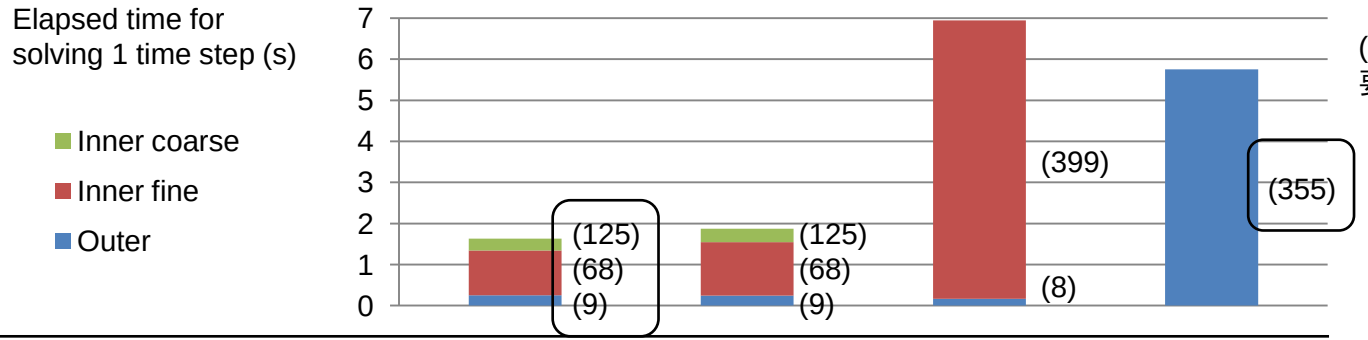
Element size: 1.0m

Material properties

Layer	1	2
Pressure wave velocity (m/s)	700	2,100
Shear wave velocity (m/s)	100	700
Density (kg/m^3)	1,500	2,100
Damping	0.25 (h_{max})	0.05
Strain Criteria	0.007	-

アルゴリズムの効果

- 京全系を使ったテスト (82,944 ノード, 270億自由度)
  - 従来法(PCG法)と比べて,「反復一回当たりの計算時間短縮」+「トータルの反復回数が少なくなる」ことで, 3.53倍の高速化



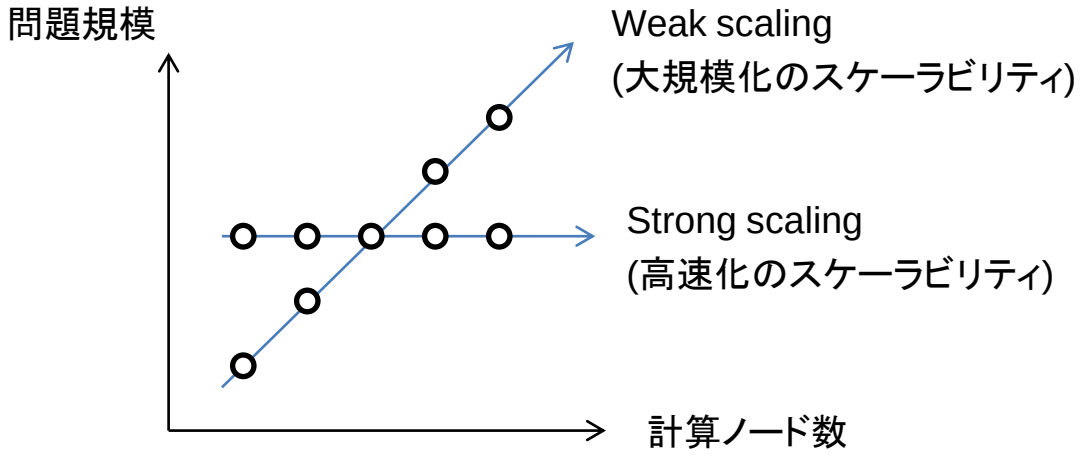
() : 収束までに要した反復回数

	GAMERA	Without mixed precision	Without mixed precision and multigrid	PCG method*
Elapsed time (s)	1.63	1.87	6.95	5.75

*3x3のブロックヤコビスケーリングを前処理に使用したCGソルバー

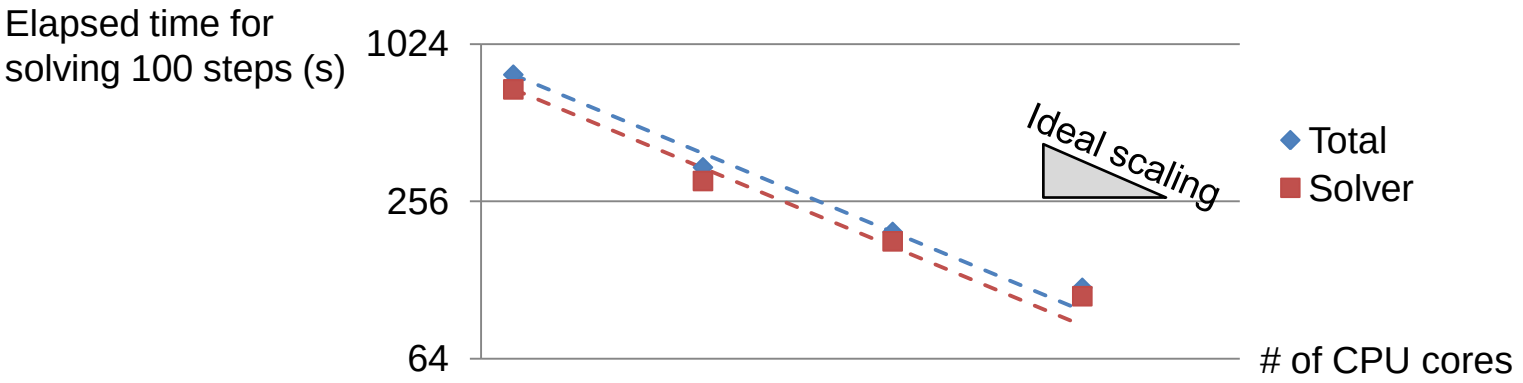
大規模・高速計算のための性能測定方法

- 2種類の指標で評価
- Strong scaling (高速化のスケーラビリティ)
  - 多数の計算ノードを使うことで，ある問題をどれだけ速く解くことができるか
- Weak scaling (大規模化のスケーラビリティ)
  - 多数の計算ノードを使うことで，大きな問題を効率よく解くことができるか
- 双方とも良い性能が出て初めて大規模問題を高速に解けるようになる



Strong scaling (高速化のスケーラビリティ)

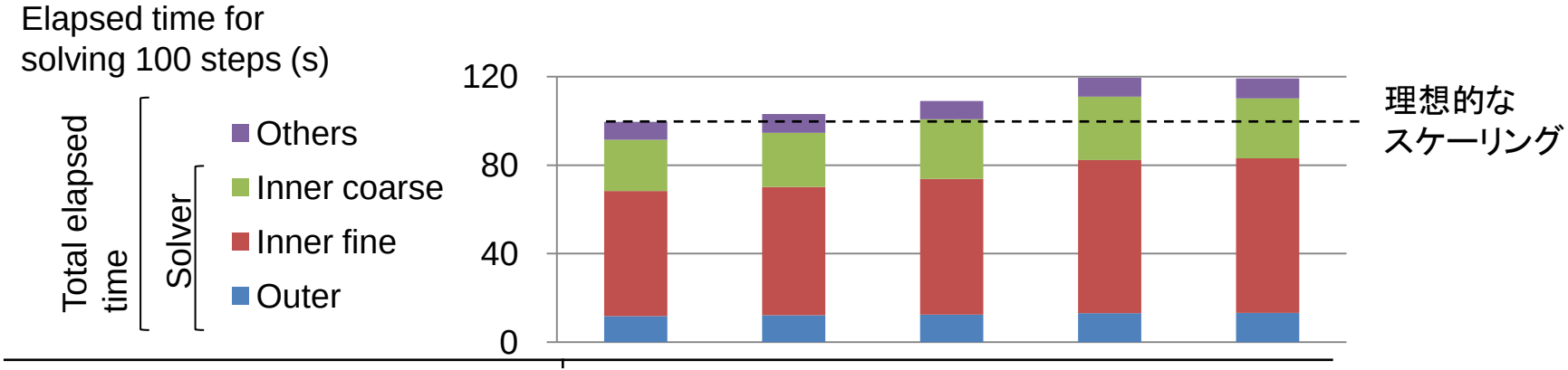
- 問題規模(120億自由度)を一定に保ったまま計算ノード数を拡大



CPUコア数 (計算ノード数)	36,864 (4,608)	73,728 (9,216)	147,456 (18,432)	294,912 (36,864)
計算ノードあたりの自由度数	2612k	1306k	653k	327k
ソルバー部の経過時間 (s)	686.5	306.3	180.0	111.0
ソルバー部の実行性能 (ピーク比%)	15.4	17.1	14.7	11.8

Weak scaling (大規模化のスケーラビリティ)

- 計算ノードあたりの問題規模を一定に保ったまま計算ノード数を拡大



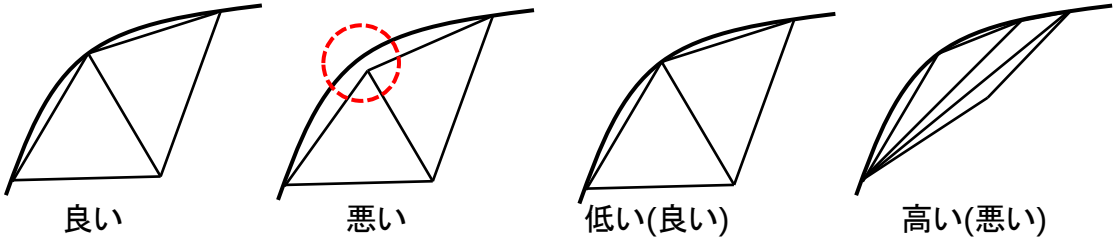
CPUコア数 (計算ノード数)	36,864 (4,608)	73,728 (9,216)	147,456 (18,432)	294,912 (36,864)	663,552 (82,944)
自由度数	15億	30億	60億	120億	270億
CPUコアあたりの自由度数	41k	41k	41k	41k	41k
ソルバー部の実行性能 (ピーク比%)	14.49	14.06	13.28	11.82	12.34

実問題のモデリング

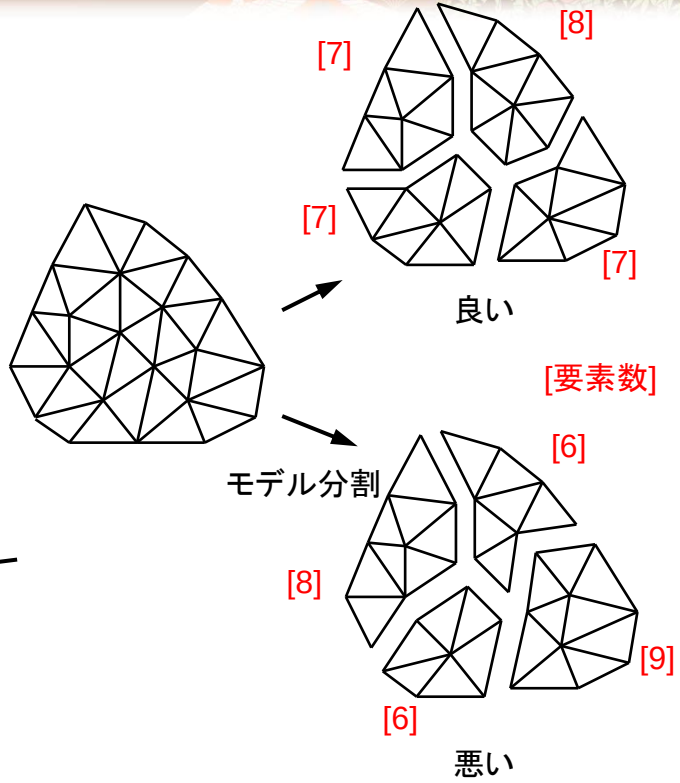
- 実問題の計算には計算プログラムの他に，計算モデル(有限要素法メッシュ)が必要
- 都市の地盤解析には100億自由度規模のメッシュが必要に
  - 高い品質でメッシュ生成・並列計算用の領域分割を行うには工夫が必要

大規模有限要素モデルに必要な性能

- 形状再現性が高い
  - 対象問題の形状を正確に離散化しているか
- 要素のアスペクト比が低い
  - 偏平な要素があると解の収束性が悪くなる
- ロードバランスが良い
  - 並列計算をするためにモデルを分割する
  - 分割後のモデルサイズ(要素数・節点数)にばらつきがあると並列計算の効率が落ちる



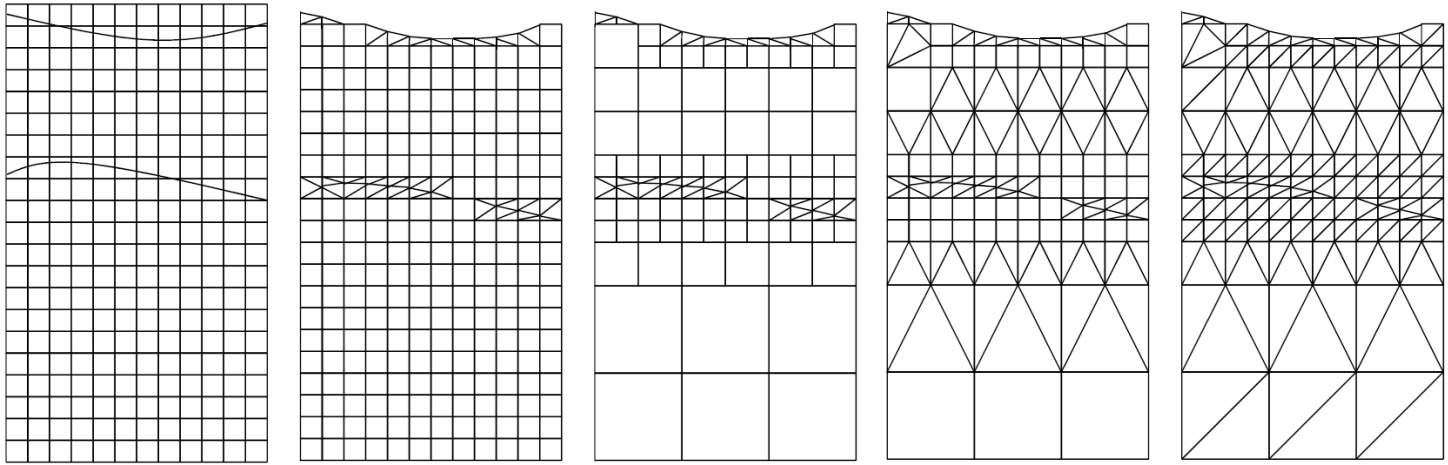
形状再現性　　アスペクト比



ロードバランス

大規模有限要素モデルの作成方法

- Background cell法ベースのモデリング手法を開発
  - セル毎にメッシュ分割を行うため，ロバストに高品質なメッシュを生成可能
  - 並列処理にも向いているため，大規模モデルを短時間で生成可能
  - 領域分割にはグラフ分割ソフトウェア(METIS)を利用



Step 1:
Overlap domain with structured cells

Step 2:
Generate high quality elements in each cell

Step 3:
Merge cells in each Octree

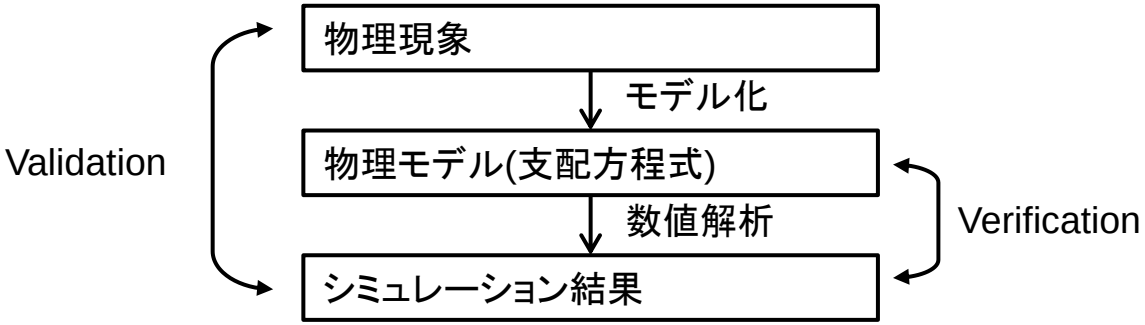
Step 4:
Make transition elements between Octrees

Step 5:
Decompose cells into tetrahedral elements

高速・大規模解析でできること

- ただ計算しただけのシミュレーション結果は実用には使えない
  - シミュレーションの実用には結果の信頼性確認が不可欠

- Verification and Validation: シミュレーションの信頼性確認方法の枠組み
  - Verification: 物理モデルが正確に解かれているか,
    Validation: シミュレーション結果が実際の現象に一致しているか,
    に分けてシミュレーション結果の確からしさを検証する



An Overview of the PTC 60 / V&V 10, Guide for Verification and Validation in Computational Solid Mechanics, *ASME*, http://cstools.asme.org/csconnect/pdf/CommitteeFiles/24816.pdf

高速･大規模解析でできること

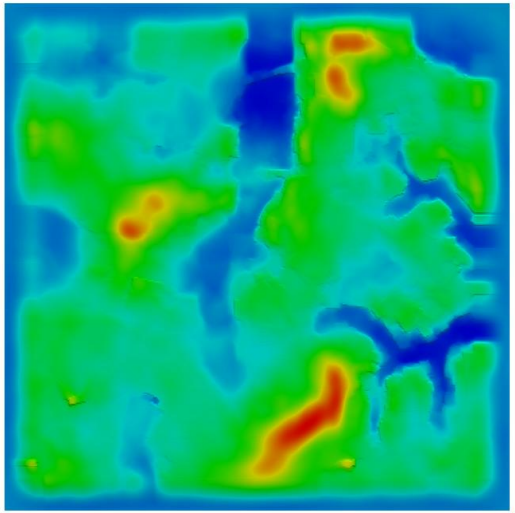
- 従来は，解析領域の一部を取り出した準備解析により，メッシュサイズを決めていた
  - 局所的に揺れが増幅する箇所では収束が甘くなる可能性

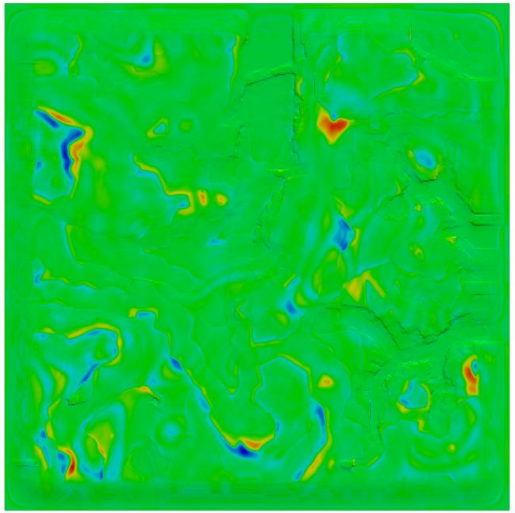
- 同じ数理問題に対して，分解能を変えた解析モデルを作成し，解析領域全体の解の収束性を確かめる

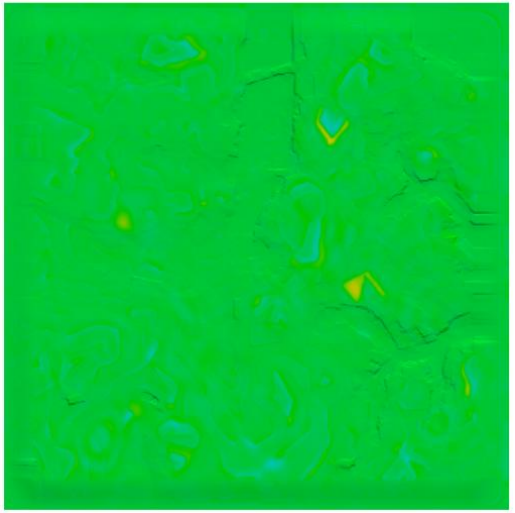
要素サイズ	自由度数
0.66m	107億
1.0m	33億
2.0m	4.7億

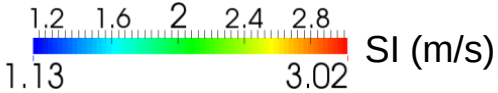
解の収束性確認

- 要素サイズ0.66mのモデルを使うことで，地表面での地震動分布は1%以内の誤差で収束

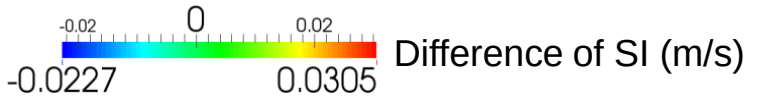








0.66m model

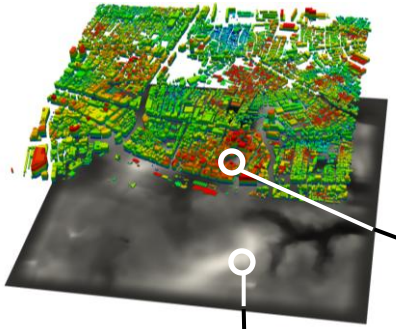


0.66m model – 2m model

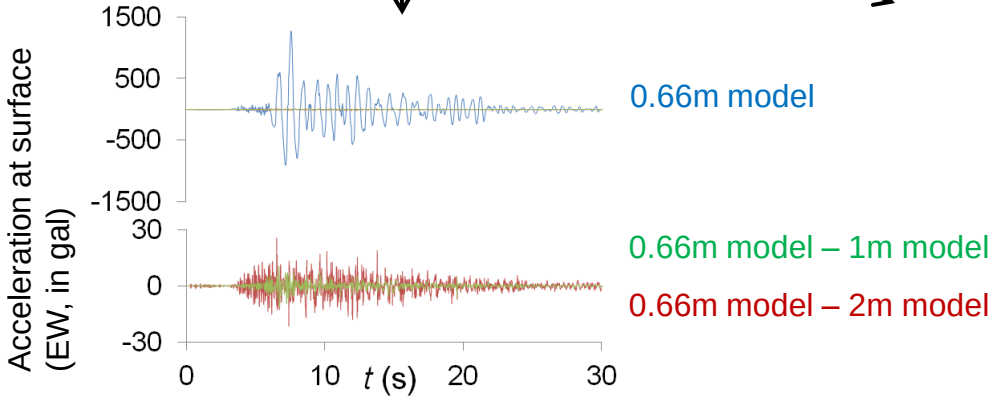
0.66m model – 1m model

※SI値：地震動の強さの指標で，構造物の被害に相関がある

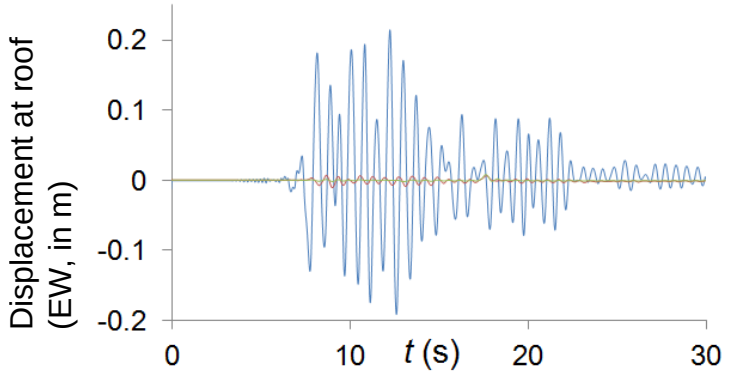
解の収束性確認



- 時刻歴波形も1%以内の誤差で収束
- 京＋GAMERAによって初めて都市スケールで実現



Convergence of ground acceleration



Convergence of structure response

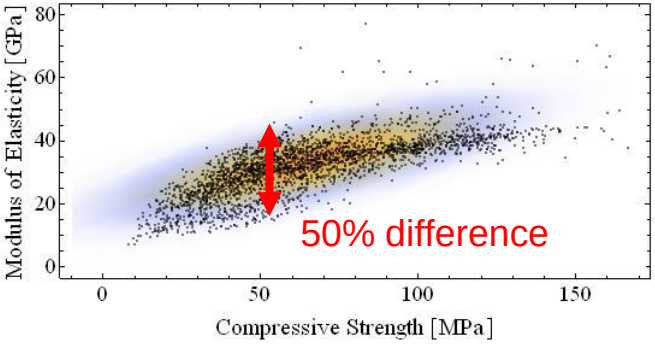
今後に向けて：統合地震シミュレーション

- 統合地震シミュレーション：地震シミュレーション手法を組み合わせることで，断層から都市応答まで一貫して解析する
  - 地盤震動解析を他の地震解析手法と組み合わせることで，地震防災の高度化を目指す

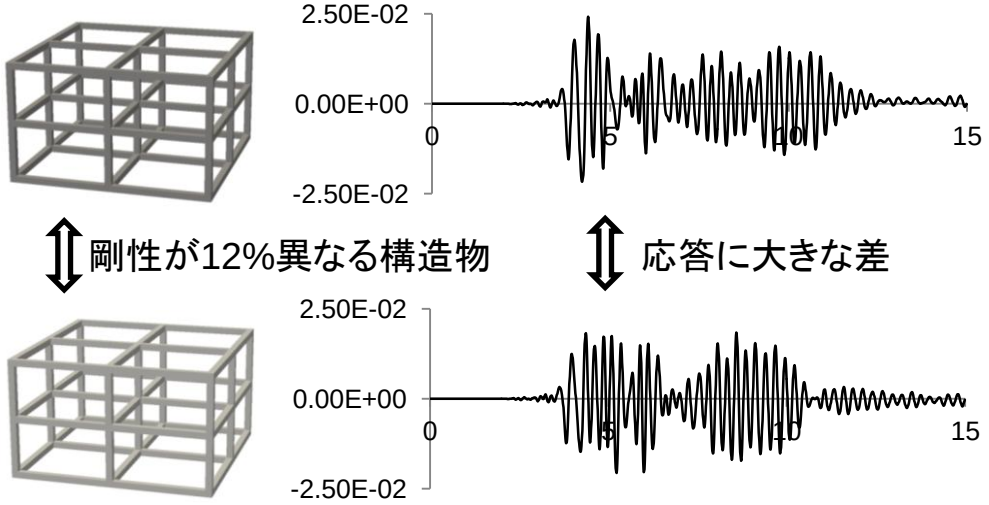
- 今回紹介する試み
  - 構造物解析の多数ケース解析との組み合わせによるstochastic simulation

多数ケースの構造物解析による
Stochastic simulation

- 実際の都市データには不確実性がある

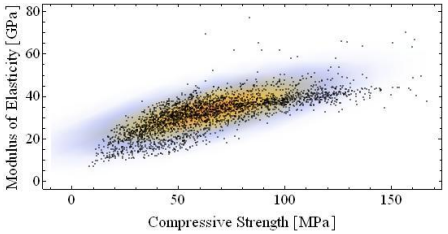


コンクリートの強度・剛性分布

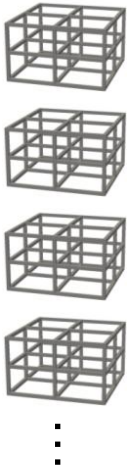


多数ケースの構造物解析による
Stochastic simulation

- 実際の都市データには不確実性がある

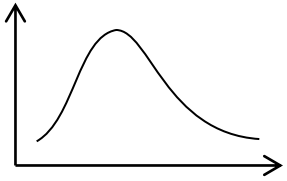


入力分布



入力分布に沿う
モデル群

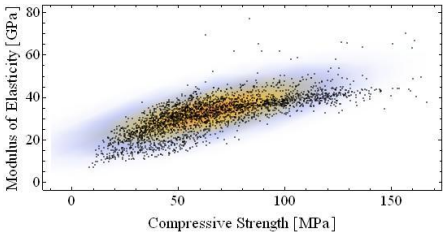
被害確率



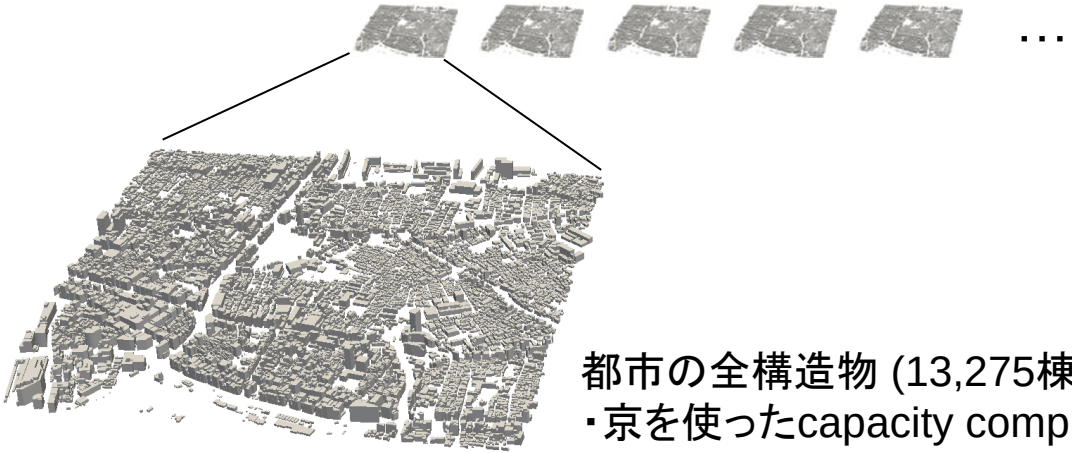
多数のモデルの応答を計算することで，
被害の確率分布を求める

多数ケースの構造物解析による
Stochastic simulation

- 実際の都市データには不確実性がある



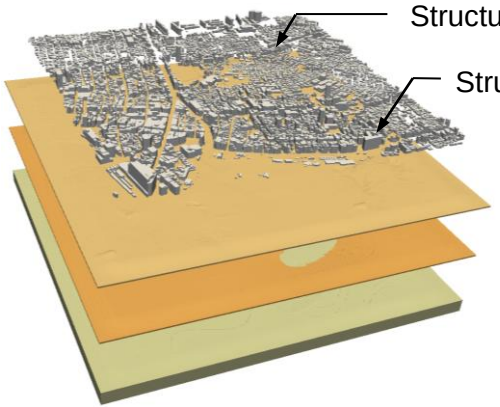
入力分布

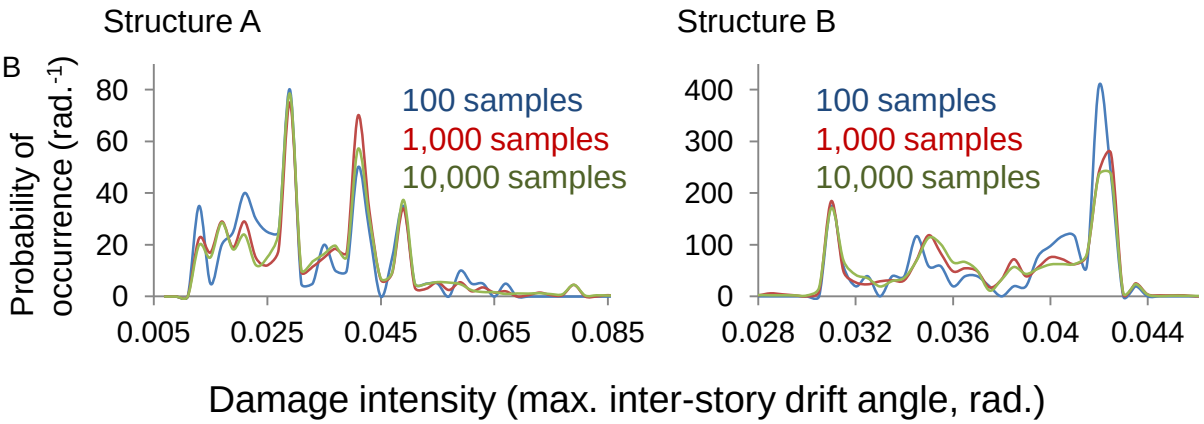


都市の全構造物 (13,275棟)に適用
・京を使ったcapacity computing

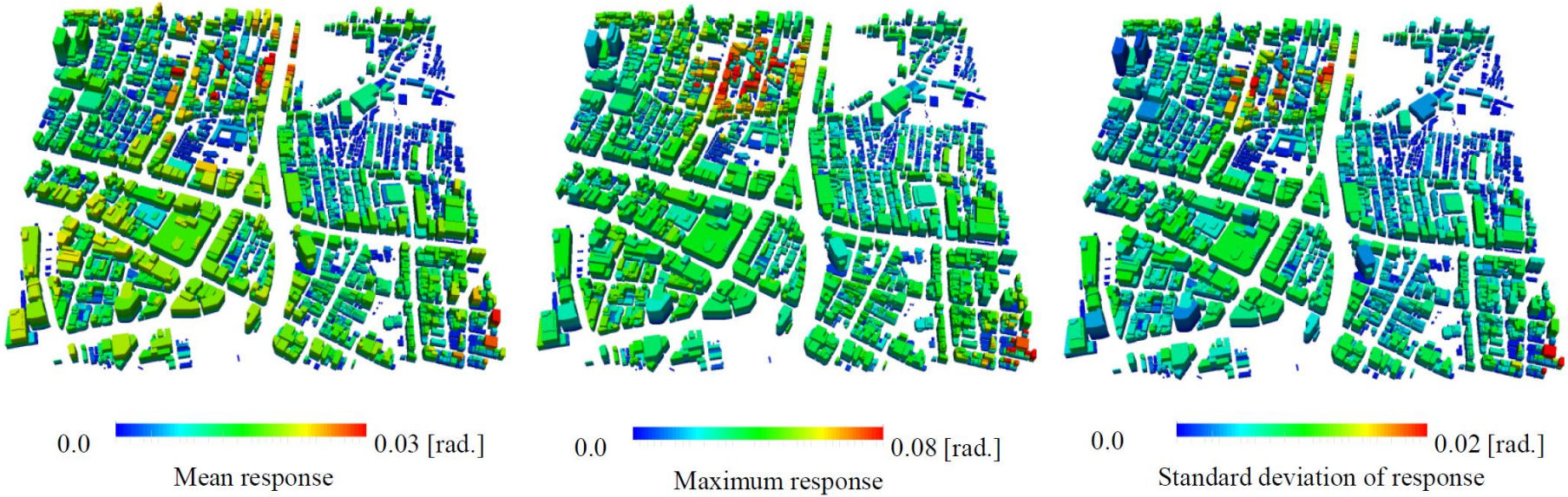
多数ケースの構造物解析による
Stochastic simulation

- 被害の確率分布を求めるためには多数のサンプルが必要
  - サンプル数を100 → 1,000 → 10,000サンプルに増やす
  - 10,000サンプルにて応答が収束
  - 京コンピュータ80,000 CPUコアを3h 56min使用して計算





多数ケースの構造物解析による
Stochastic simulation



Stochastic features of damage probability distribution

都市の全構造物の被害確率分布を計算
→ 単一ケースの解析に比べて被害推定の信頼性向上が期待できる

まとめ

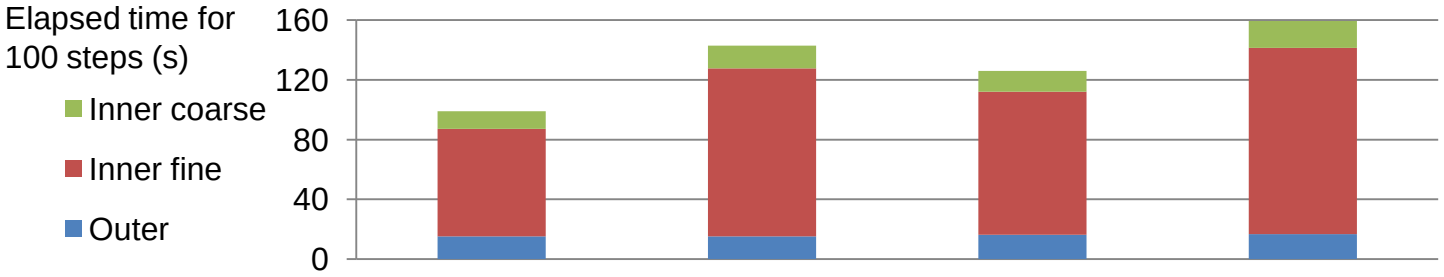
- 地盤解析：地震シミュレーションの信頼性向上のために重要な問題
  - 構造物や人的被害の評価に重要
  - 複雑形状と非線形性のため，大規模な有限要素解析が必要に

- 高速化によってできるようになったこと
  - 100億自由度の大規模地盤震動シミュレーション
  - 実都市スケールの地盤震動解析の収束性解析

- 他の地震分野の計算手法と組み合わせることで，地震防災の高度化が期待できる
  - 多数ケース解析による構造物のstochastic simulation

今後の展望

- GAMERAの計算の大部分は単精度演算
  - 単精度演算の高速な計算機でより効果的と期待
  - 現在GPUに移植中



Compute environment	Stampede 256 nodes (each with dual Intel Xeon E5-2680 CPUs)		K computer 256 nodes (each with single SPARC64VIIIfx CPU)	
Solver	GAMERA	GAMERA (double precision)	GAMERA	GAMERA (double precision)
Elapsed time (s)	99.0	142.9	125.9	159.5
Ratio between GAMERA and GAMERA (double precision)	**1.44**	←	1.27	-

※SPARC64VIIfx CPUの単精度・倍精度のピーク演算性能は同じ，Xeon E5-2680CPUの単精度のピーク演算性能は倍精度の2倍