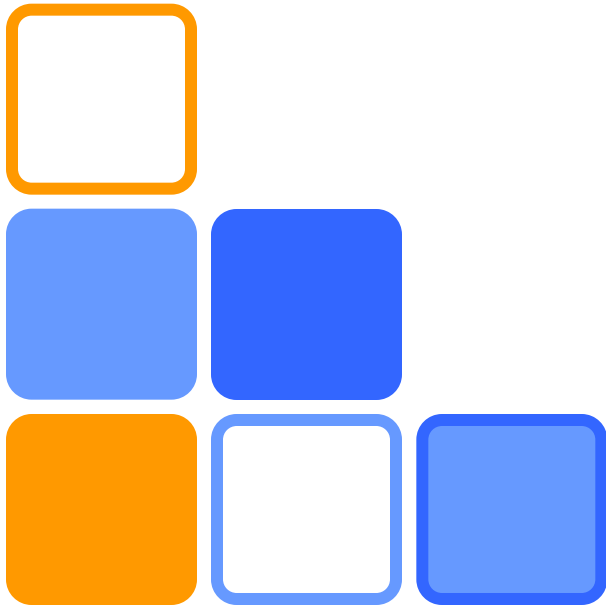


Scilabにおける疎行列向け高精度演算



東京理科大学大学院 理学研究科
数理情報科学専攻 修士2年
吉川慧子

東京理科大学大学院 齊藤翼
東京理科大学 石渡恵美子
筑波大学 長谷川秀彦



研究概要

□ 背景

浮動小数点演算では丸め誤差の影響は避けられず、演算結果の検証にはより高精度な演算が必要

□ 目的

誰もが、簡単に、手間なく 使える
高精度演算環境の構築

➤ 手段

フリーの数値計算ソフトScilab上で開発

- ✓ Matlabライクでプログラムが簡単
- ✓ 型定義, オーバーロードが可能



高精度演算に対する取り組み

□ QuPAT (Quadruple Precision Arithmetic Toolbox) [1]

Scilab Toolbox Japan Contest 2009 最優秀賞(学生部門)受賞

- 倍精度・4倍精度演算が同時に使える



□ MuPAT (Multiple Precision Arithmetic Toolbox) [2]

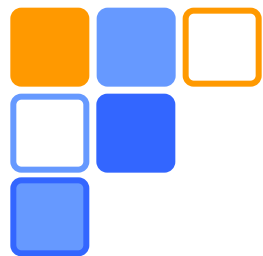
Scilab Toolbox Japan Contest 2011 最優秀賞(学生部門)受賞

- 8倍精度へ拡張
- C言語による外部関数を利用し高速化

疎行列データ型にも対応させたい

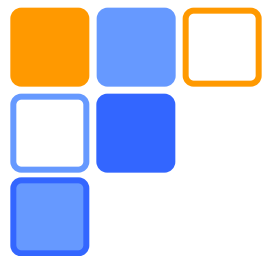
[1] T. Saito, E. Ishiwata and H. Hasegawa, Development of quadruple precision arithmetic toolbox QuPAT on scilab, ICCSA2010, Proceedings Part II, (2010).

[2] S. Kikkawa, T. Saito, E. Ishiwata and H. Hasegawa, Development and acceleration of multiple precision arithmetic toolbox MuPAT for Scilab , JSIAM Letters. (in revision)



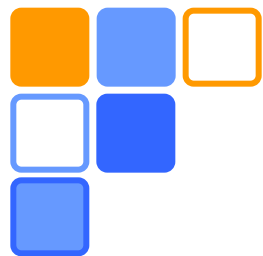
目次

- はじめに
- 高精度演算ツールボックスMuPAT
- MuPATに対する疎行列向け高精度演算の実装
- 疎行列向け高精度演算の有効性検証実験
- まとめ



目次

- はじめに
- 高精度演算ツールボックスMuPAT
- MuPATに対する疎行列向け高精度演算の実装
- 疎行列向け高精度演算の有効性検証実験
- まとめ



Double-Double(DD), Quad-Double(QD)

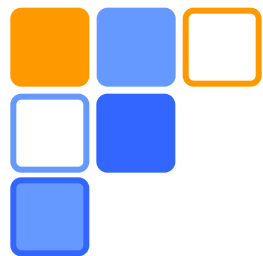
DD: 2つの倍精度数で4倍精度数を表現する手法

QD: 4つの倍精度数で8倍精度数を表現する手法

QDの数 $A = a_0 + a_1 + a_2 + a_3$

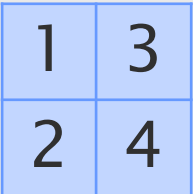
$$\left(\begin{array}{l} a_i : \text{倍精度数} \\ |a_{i+1}| \leq \frac{1}{2} \text{ulp}(a_i), \quad (i = 0, 1, 2) \\ \text{※ulp (units in the last place)} \end{array} \right)$$

演算は倍精度の四則演算のみで実装



Scilabにおける倍精度変数の扱い

スカラー, ベクトル, 行列はすべて
constant型として扱われる

	constant	
	code	size
スカラー	<code>a = 1;</code>	1×1 
ベクトル	<code>a = [1;2];</code>	2×1 
行列	<code>a = [1,3;2,4];</code>	2×2 



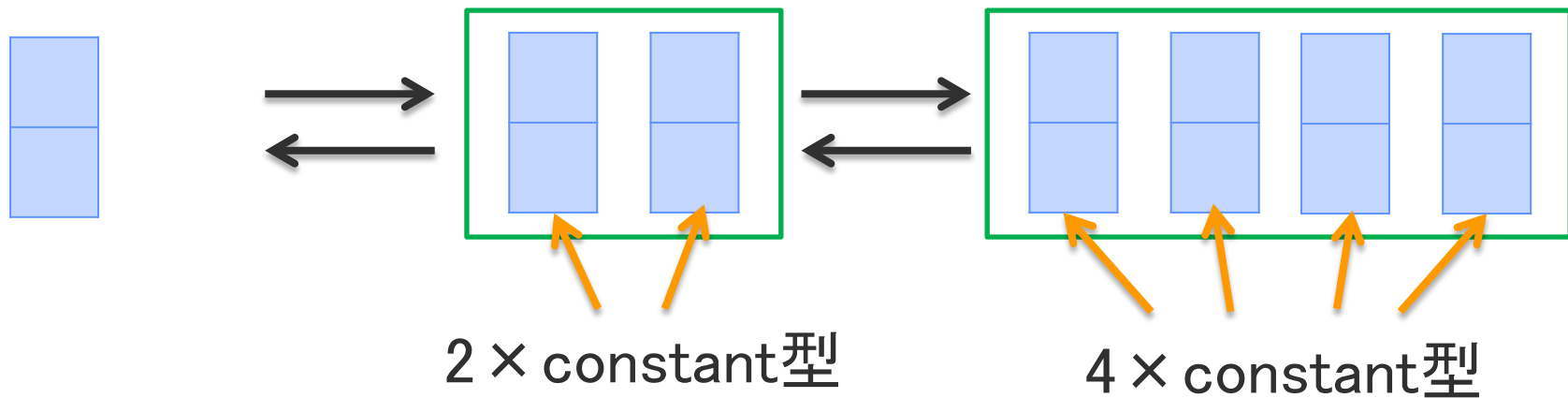
MuPATにおけるデータ型の拡張

constant型の組を1つのデータ型として定義

constant型

4倍精度 dd型

8倍精度 qd型

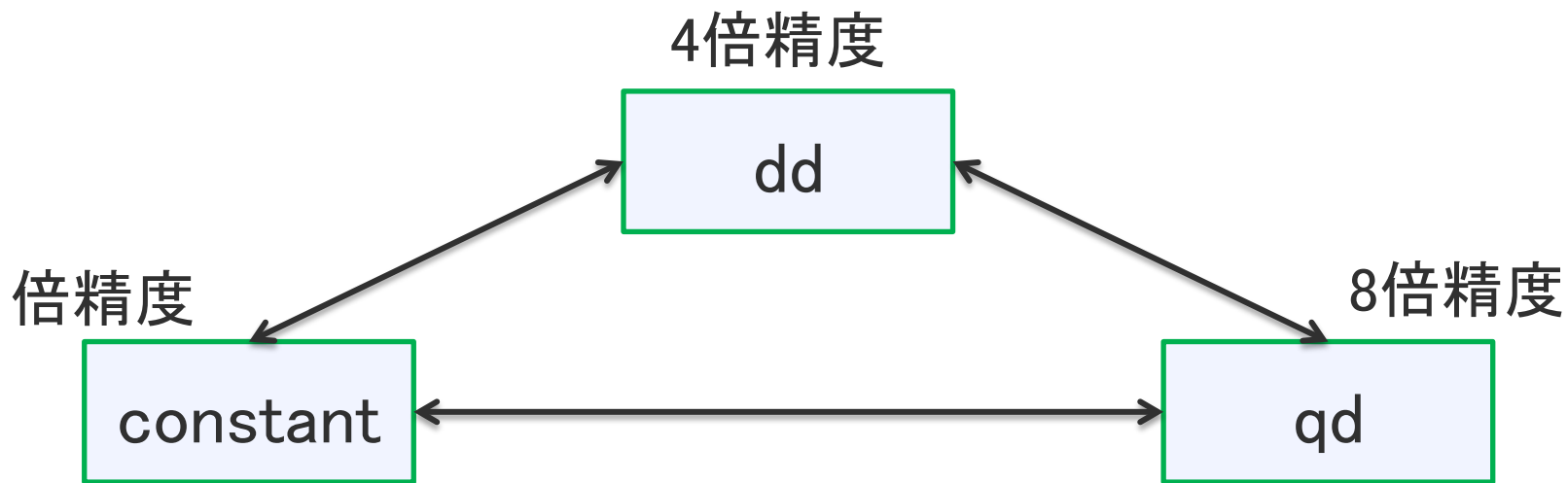


dd型, qd型ともにベクトルや行列も扱える



MuPATにおけるデータ型の拡張

MuPATでは、倍精度, 4倍精度, 8倍精度演算で
同じ演算子 $+$, $-$, $*$, $/$ を使用



演算子 $+$, $-$, $*$, $/$ が定義されている



dd型, qd型で使える関数

関数名	用途
<code>abs(a)</code>	絶対値
<code>norm(a,n)</code>	aのnノルム ($n=1,2,\text{inf}$)
<code>nrt(a,n)</code>	aのn乗根
<code>sqrt(a)</code>	平方根
<code>sin(a)</code>	正弦
<code>cos(a)</code>	余弦
<code>tan(a)</code>	正接

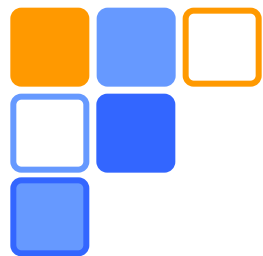
関数名	用途
<code>ceil(a)</code>	切り上げ
<code>floor(a)</code>	切り下げ
<code>lu(a)</code>	LU分解
<code>qr(a)</code>	QR分解
<code>a(i,j)</code>	成分の取出
<code>a(i,j) = b</code>	成分への代入
<code>a'</code>	行列aの転置

constantと同じ記述で使用



その他に実装した関数

constant型	dd型	qd型	用途
zeros(m,n)	ddzeros(m,n)	qdzeros(m,n)	零行列の生成
eye(m,n)	ddeye(m,n)	qdeye(m,n)	単位行列の生成
rand(m,n)	ddrand(m,n)	qdrand(m,n)	乱数の生成
—	ddip(x,y)	qdip(x,y)	内積
—	ddnormF(a)	qdnormF(a)	フロベニウスノルム
—	ddpi()	qdpi()	円周率(定数)
—	ddprint(a)	qdprint(a)	10進による出力
—	ddinput(a)	qdinput(a)	10進による入力



MuPATの特徴

- 倍精度, DD, QD演算で**同じ演算子**を使える



高精度演算への切り替えが簡単

- 倍精度, DD, QD演算を**同時に**使える



部分的な高精度化, 混合精度演算ができる

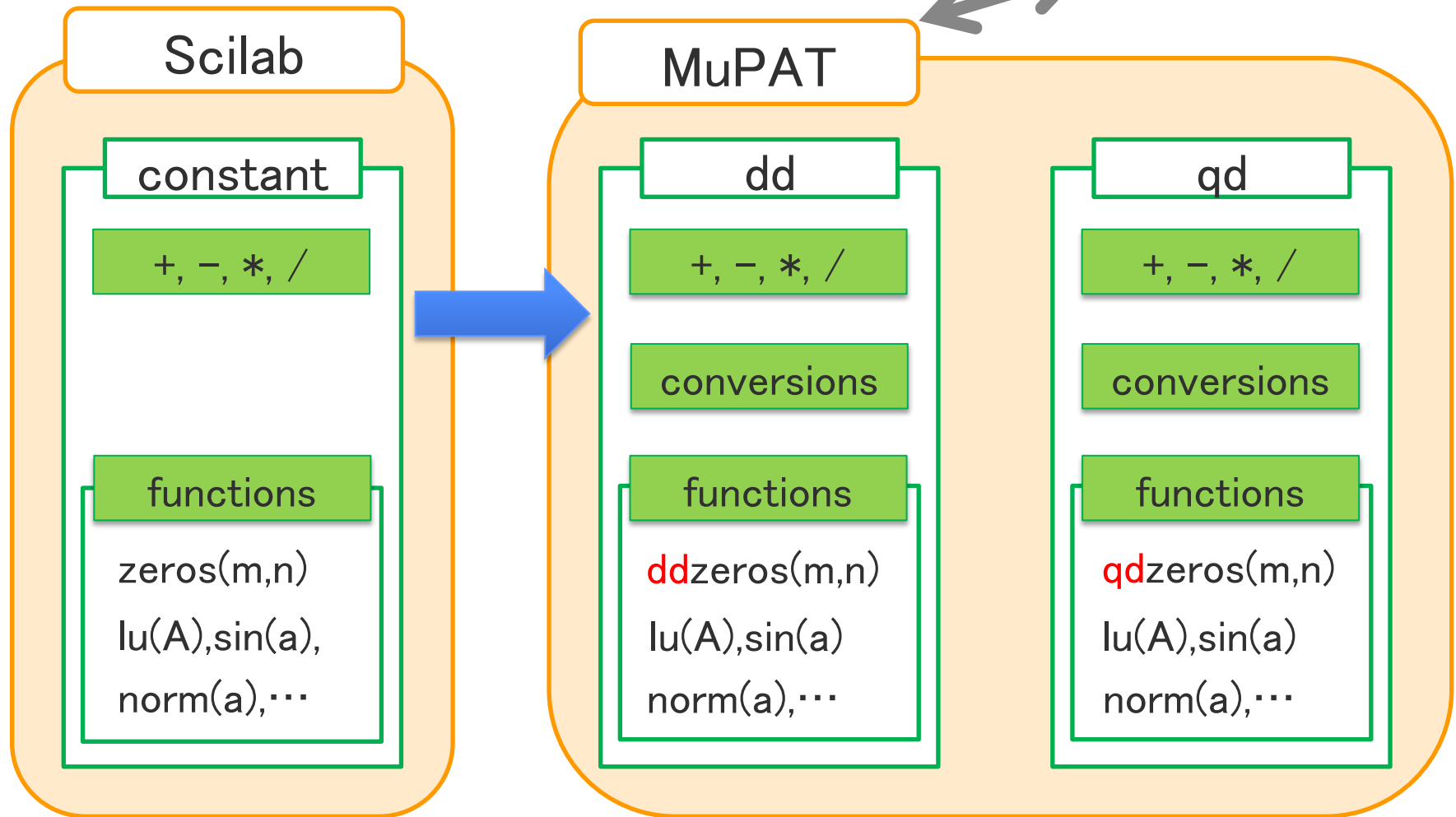
- Scilabのみの実装と外部関数による実装を選択可能

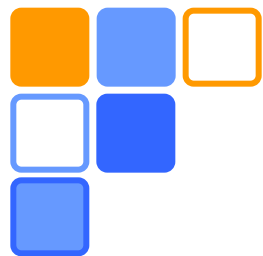
- Scilabのみの実装: ハードウェアやOSに**依存しない**
- 外部関数を利用した実装: 高速なQD演算が可能

高精度演算を簡単に利用できる有用なツール

ScilabとMuPATの関係

外部関数
(Cプログラム)





目次

- はじめに
- 高精度演算ツールボックスMuPAT
- MuPATに対する疎行列向け高精度演算の実装
- 疎行列向け高精度演算の有効性検証実験
- まとめ



MuPATの現状

□ MuPATは密行列データ型のみ対応

➤ 密行列データ型

- ✓ 定義できる行列は数千次元程度
メモリ4GBの場合, 定義できる行列は
DDで最大約 4000 次元
QDで最大約 2500 次元

増やしたい

✓ 演算の実行時間

例: Ax を100回反復した時の実行時間 604秒

$\left[\begin{array}{l} A : \text{倍精度の4000次元乱数行列} \\ x : \text{8倍精度の4000次元乱数ベクトル} \end{array} \right]$

短縮したい

MuPATの高精度演算を
疎行列データ型に対応させたい



MuPATの疎行列への対応

□ 疎行列向け高精度演算の実装方針

1. 型の定義

Scilabの倍精度疎行列データ型sparseを利用して,
4倍精度疎行列データ型 **ddsp**
8倍精度疎行列データ型 **qdsp** を新たに定義

2. 演算の定義

sparse, ddsp, qdsp に関わる全ての演算を定義

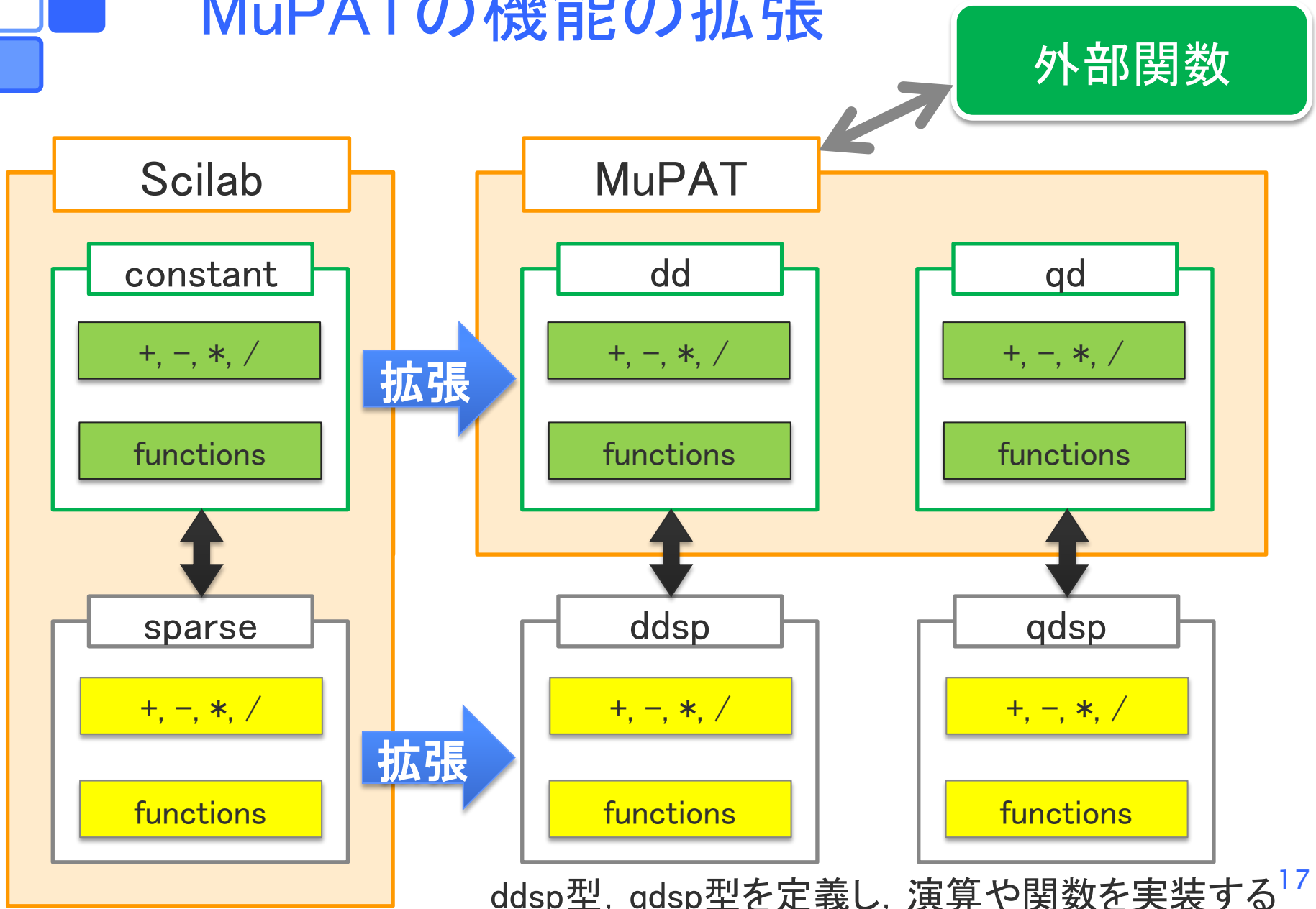
例: sparse * dd, dd * ddsp, ddsp * qdsp .等.

(疎 * 密) (密 * 疎) (疎 * 疎)

3. 関数の定義

sparse用関数をDD,QDへ拡張

MuPATの機能の拡張





Scilabのsparse型

COO形式

(Coordinate list)

	2		5
1			
		4	
	3		

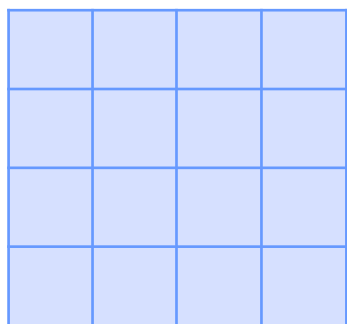
```
a =  
( 4, 4) sparse  
matrix  
( 1, 2)      2.  
( 1, 4)      5.  
( 2, 1)      1.  
( 3, 3)      4.  
( 4, 2)      3.  
row column value
```

□ sparse型の特徴

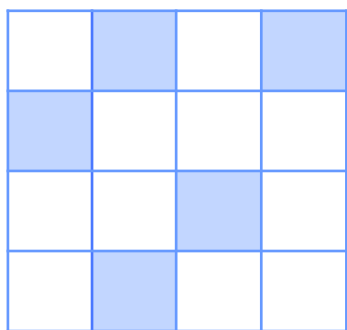
- 四則演算子 (+, -, *, /) を利用できる
- メモリを節約できる
 - ✓ 非零要素率5% → 密行列(constant型)の約7.5%

データ型の拡張

double



constant型

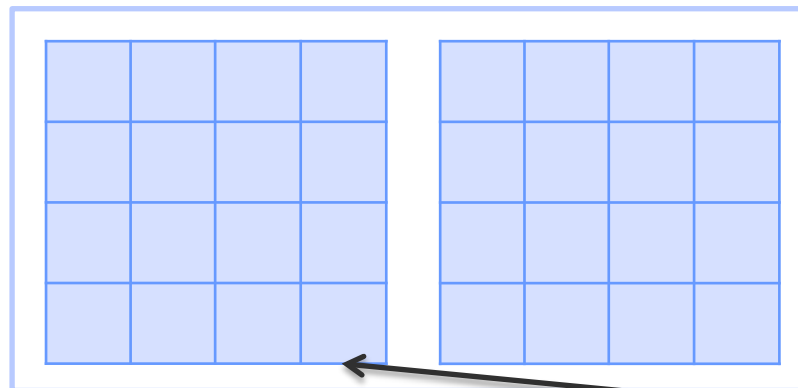


sparse型

拡張



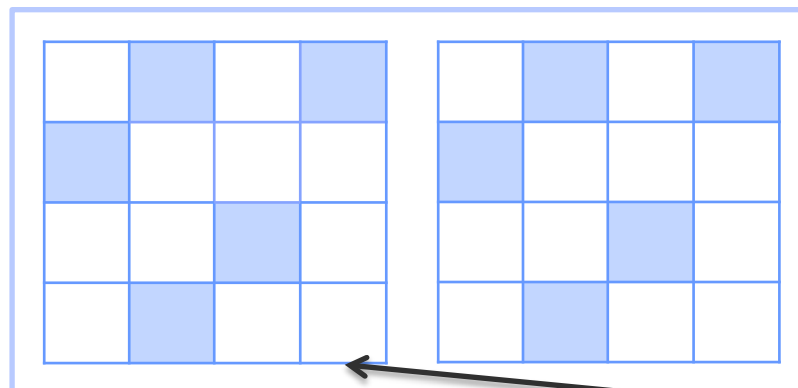
DD



dd型

constant型

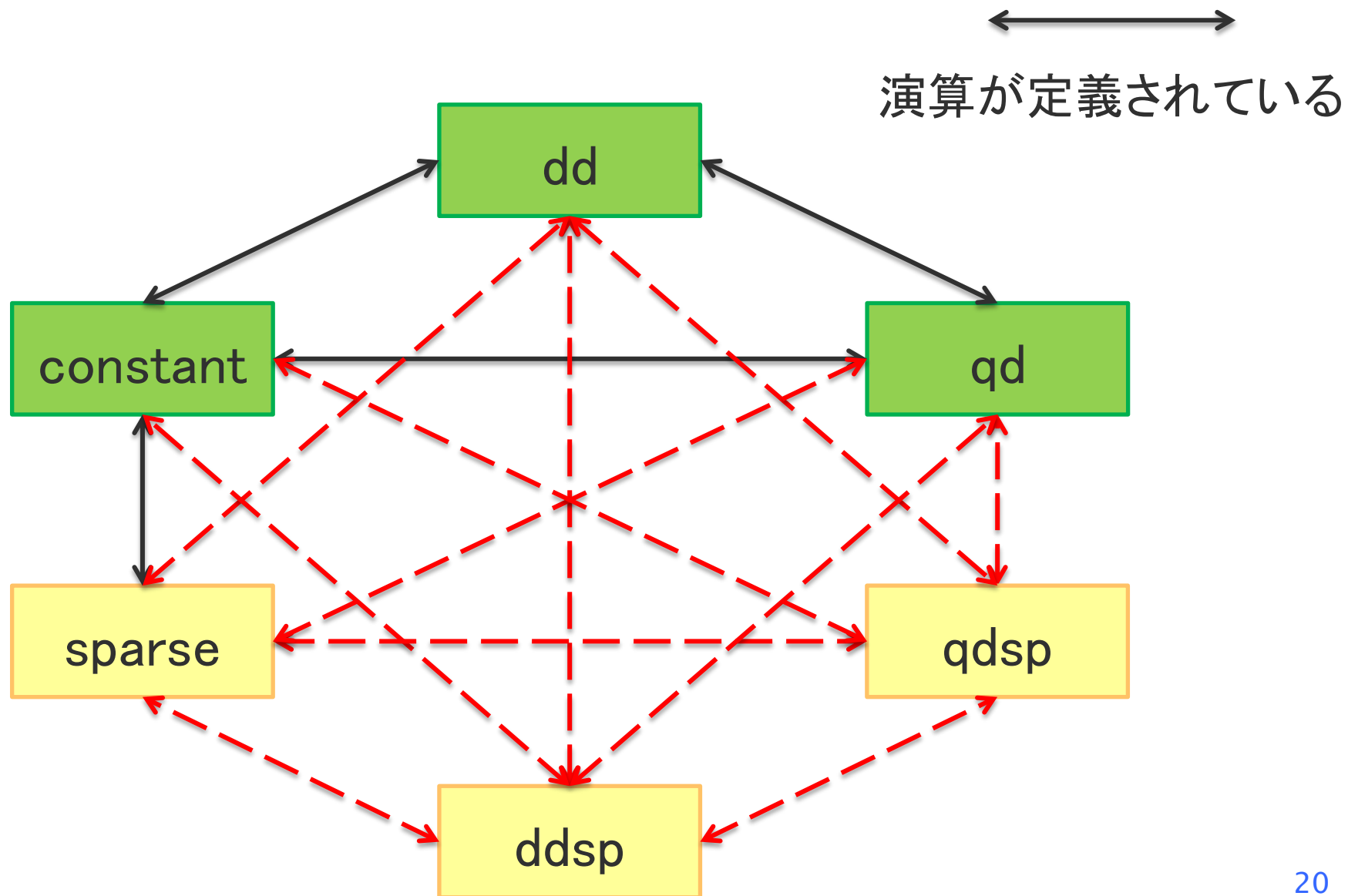
拡張

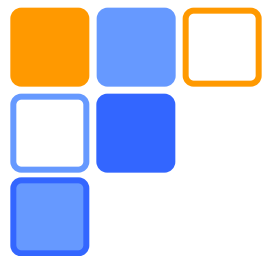


ddsp型

sparse型 19

演算の拡張方針





MuPATの外部関数を利用した実装

□ MuPATでは2種類の実装を用意

➤ Scilabのみの実装

- ✓ OSやハードウェアに依存しない

➤ C言語による外部関数を利用した実装

- ✓ 高速な演算が可能

➤ 日本応用数理学会「行列・固有値の解法とその応用」研究部会 第12回研究会（2011年11月国立情報学研究所）

- 高速なScilab用高精度演算環境の実装（吉川他）



外部関数による高速化

各四則演算を 10^6 回反復

単位は秒

		$+$ / $-$	$\times$	$\div$
d	Num. of double	1	1	1
	MuPAT	0.016	0.014	0.013
DD	Num. of double	11	24	27
	MuPAT	0.21 (12.8)	0.39 (28.4)	0.39 (30.6)
	MuPAT with C	0.26 (16.4)	0.31 (22.8)	0.32 (24.9)
QD	Num. of double	91	217	649
	MuPAT	2.91 (181.7)	4.21 (309.7)	21.29 (1663.5)
	MuPAT with C	0.34 (21.1)	0.39 (28.3)	0.39 (30.3)

※括弧内は倍精度演算に対する倍率

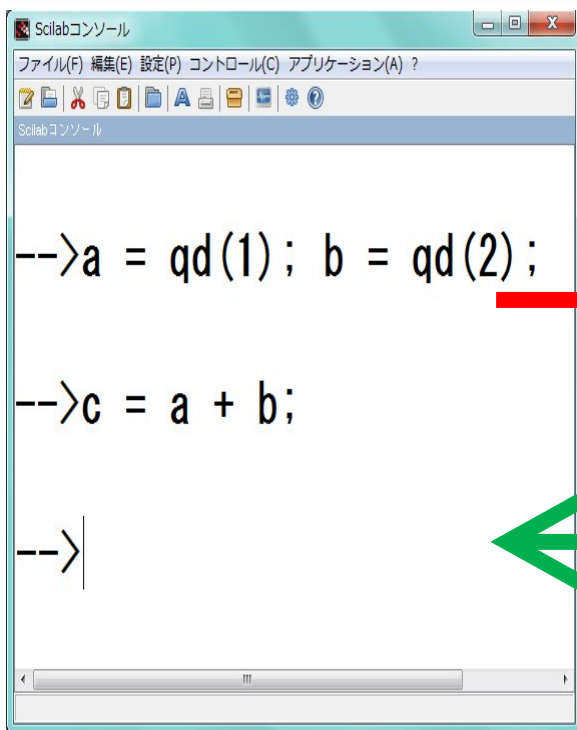
※計算環境 Intel Core i5 2.5GHz, 4GB, Windows 7, Scilab version 5.3.3

外部関数を利用した実装

Scilabコンソール

Scilabソースコード

外部関数
(Cプログラム)



```
Scilabコンソール
ファイル(F) 編集(E) 設定(P) コントロール(C) アプリケーション(A) ?
Scilabコンソール

-->a = qd(1); b = qd(2);
-->c = a + b;
-->|
```

```
function %qd_a_qd
```

外部関数
の
呼び出し

```
endfunction
```

QD演算

□ 外部関数を用いることで高速な実行が可能

Scilab上での加減算の実装

□ 加算/減算

実装する関数	定義される演算
%sp_a_qd	sp + qd
%qd_a_sp	qd + sp
%qdsp_a_qdsp	qdsp + qdsp
%qdsp_a_ddsp	qdsp + ddsp
%ddsp_a_qdsp	ddsp + qdsp
%qdsp_a_sp	qdsp + sp
%sp_a_qdsp	sp + qdsp
%qdsp_a_qd	qdsp + qd
%qd_a_qdsp	qd + qdsp
%qdsp_a_dd	qdsp + dd
%dd_a_qdsp	dd + qdsp
%qdsp_a_s	qdsp + double
%s_a_qdsp	double + qdsp
%ddsp_a_qd	ddsp + qd
%qd_a_ddsp	qd + ddsp

- QD疎行列加算 $A + B$ を
1000回反復したときの実行時間
〔 A, B : qdspの1000次元疎行列
(非零要素率5%)〕

Scilabのみ・・・ 47.65 秒

外部関数利用・・・ 92.35 秒

➡ 外部関数是不利用しない

- 密の場合と同じアルゴリズムで
実装可能

Scilab上での乗算の実装

乗算

実装する関数	定義される演算
%sp_m_qd	sp * qd
%qd_m_sp	qd * sp
%qdsp_m_qdsp	qdsp * qdsp
%qdsp_m_ddsp	qdsp * ddsp
%ddsp_m_qdsp	ddsp * qdsp
%qdsp_m_sp	qdsp * sp
%sp_m_qdsp	sp * qdsp
%qdsp_m_qd	qdsp * qd
%qd_m_qdsp	qd * qdsp
%qdsp_m_dd	qdsp * dd
%dd_m_qdsp	dd * qdsp
%qdsp_m_s	qdsp * double
%s_m_qdsp	double * qdsp
%ddsp_m_qd	ddsp * qd
%qd_m_ddsp	qd * ddsp

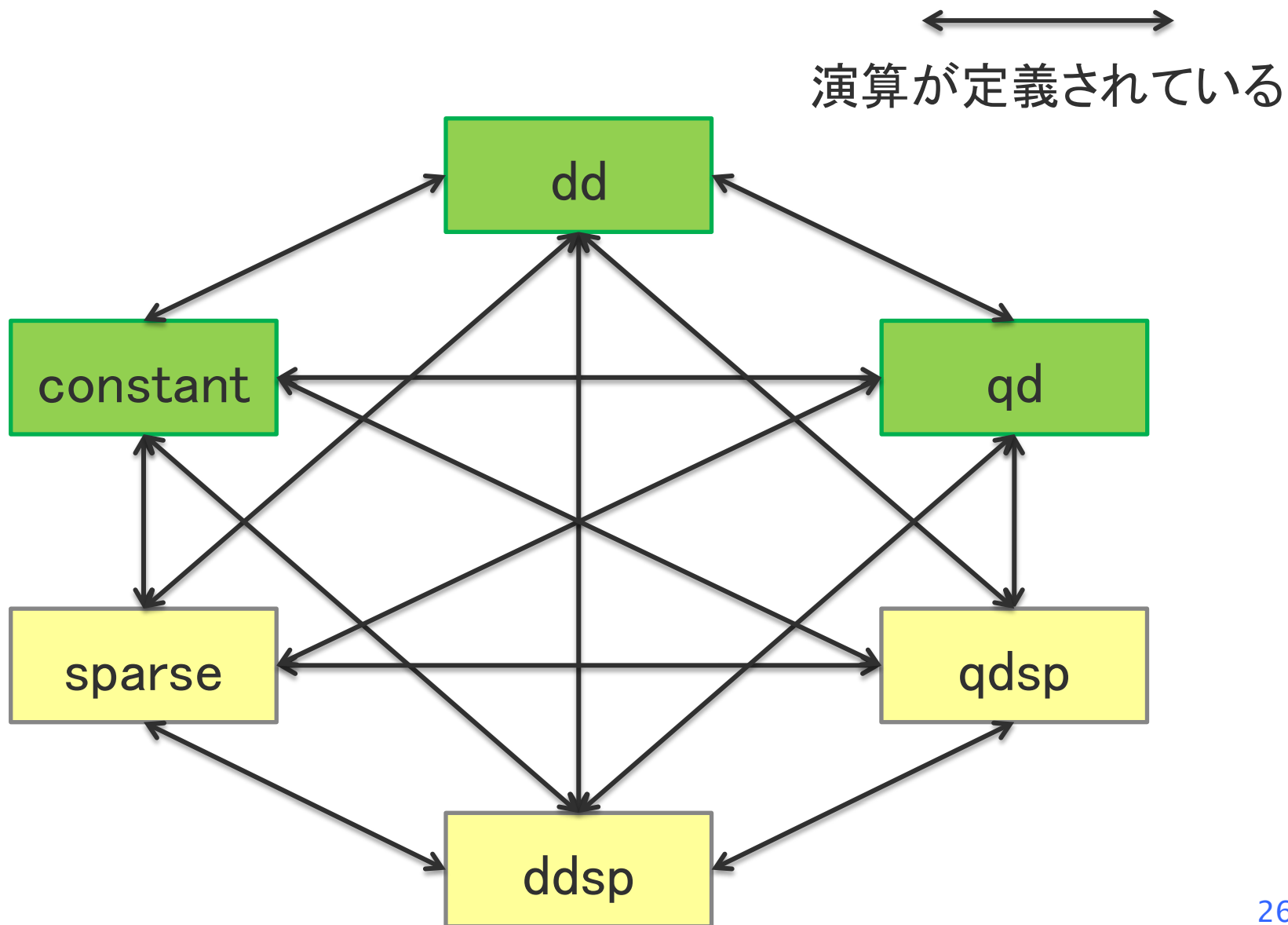
➤ 行列ベクトル積 Ax を
1000回反復したときの実行時間
(A : ddspの1000次元疎行列
(非零要素率5%)
 x : dd の1000次元ベクトル)

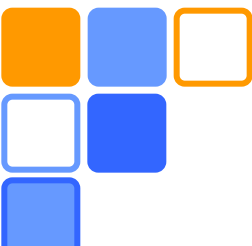
Scilabのみ・・・ 1135.78 秒
外部関数利用・・・ 10.92 秒

➡ 外部関数を利用する

➤ 疎行列同士の演算のみ
文献[4]の方法を利用

演算の拡張





疎行列向け関数の拡張

sparse型	ddsp型	qdsp型	用途
sparse	ddsparse	qdsparse	型の定義
issparse	isddsparse	isqdsparse	型の判定
full	full	full	密行列型(constant, dd, qd)への変換
nnz	nnz	nnz	非零要素数
spget	spget	spget	行列番号と値の抜き出し
sp2adj	ddsp2adj	qdsp2adj	隣接形式への変換
adj2sp	adj2ddsp	adj2qdsp	隣接形式からの変換
speye	ddspeye	qdspeye	疎単位行列
spones	ddspones	qdspones	成分が全て1の疎行列
sprand	ddsprand	qdsprand	疎乱数行列
spzeros	ddspzeros	qdspzeros	疎零行列
abs	abs	abs	絶対値
diag	diag	diag	対角要素の抜き出し



疎行列向け高精度演算

□ MuPATに疎行列向けの4・8倍精度演算の実装

➤ 実装方法

- ✓ Scilabのsparse型を組み合わせ
4,8倍精度疎行列データ型 `ddsp`, `qdsp` を定義
- ✓ 演算・関数を定義
 - 加減算 : Scilabのみの実装
 - 乗算 : 外部関数を利用

➤ 特徴

- ✓ 共通の演算子 `+`, `-`, `*` が使え, 型の変更が簡単
- ✓ 疎行列向け関数もほぼ網羅
- ✓ 非零要素率0.1%の場合, 定義できる行列は最大
`ddsp`: 120000次元, `qdsp`: 70000次元

MuPATの機能の拡張

外部関数

Scilab

constant

+, -, *, /

functions

sparse

+, -, *, /

functions

MuPAT

dd

+, -, *, /

functions

ddsp

+, -, *

functions

qd

+, -, *, /

functions

qdsp

+, -, *

functions



目次

- はじめに
- 高精度演算ツールボックスMuPAT
- MuPATに対する疎行列向け高精度演算の実装
- 疎行列向け高精度演算の有効性検証実験
- まとめ



疎行列向け高精度演算の有効性検証

- 以下の2つの実験において,
実行時間と使用メモリ量を図り,
密行列データ型と疎行列データ型を比較する.
 - 実験1: 行列ベクトル積による比較
 - 実験2: CG法による比較



実験1：行列ベクトル積による比較

□ 行列ベクトル積 Ax を100回反復

A ：倍精度の4000次元乱数行列（非零要素率5%）

x ：4倍または8倍精度の4000次元乱数ベクトル

A		密		疎	
		実行時間 (秒)	A のメモリ使用量 (Byte)	実行時間 (秒)	A のメモリ使用量 (Byte)
x (密)	DD	111.9	1.28×10^8	13.6	9.60×10^6
	QD	604.8		68.4	

DD・QD演算ともに、実行時間を約 $\frac{1}{10}$ に削減



実験2.CG法による比較

- 連立一次方程式 $Ax = b$ の解を求めるCG法に対し、MuPATの疎行列向け8倍精度演算を適用し、密行列向け8倍精度演算との比較を行う.

➤ 実験概要

係数行列 A を

倍精度密行列データ型と倍精度疎行列データ型で保持したときの、実行時間と使用メモリ量を計測する.
全ての演算には8倍精度演算を用いる.



実験条件

➤ 係数行列 A : MatrixMarketからn次対称疎行列を選択

no.	行列	次元	非零要素	非零要素率	条件数
1	nos4	100	347	3.47%	1.59E+03
2	nos3	960	8402	0.91%	3.77E+03
3	bcsstk16	4884	147631	0.62%	4.94E+09

- 真値 : $\mathbf{x}^* = (1, 1, \dots, 1)^T$
- 右辺項 : 倍精度演算 $\mathbf{b} = A\mathbf{x}^*$ により作成.
- 初期ベクトル : $\mathbf{x}_0 = (0, 0, \dots, 0)^T$
- 収束判定条件 : $\|\mathbf{r}_k\|/\|\mathbf{r}_0\| \leq 10^{-10}$
- 実験環境:

Intel Core i5 2.5GHz, 4GB, Windows 7, Scilab version 5.3.3



実験結果

密・8倍精度

	$\ b - Ax\ _2 / \ b\ _2$	反復回数	実行時間(秒)	使用メモリ(Byte)
nos4	5.25×10^{-11}	88	0.9	27487
nos3	8.96×10^{-11}	271	262.9	953708
bcsstk16	9.73×10^{-11}	478	8515.3	23952276

疎・8倍精度

	$\ b - Ax\ _2 / \ b\ _2$	反復回数	実行時間(秒)	使用メモリ(Byte)
nos4	5.25×10^{-11}	88	0.3	18084
nos3	8.96×10^{-11}	271	14.6	53450
bcsstk16	9.73×10^{-11}	478	241.1	522133

メモリ量削減や高速化を実現



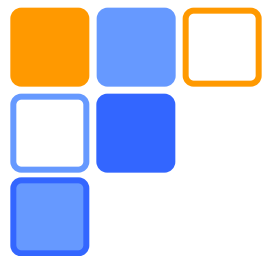
まとめ

□ MuPATに疎行列向け4・8倍精度演算を実装

- sparse型を組み合わせて
4,8倍精度疎行列データ型を定義
- 演算・関数を定義
加減算はScilabのみの実装, 乗算は外部関数を利用
 - ✓ 共通の演算子+, −, * が使える
 - ✓ 疎行列向け関数もほぼ網羅

□ 密と疎で実行時間とメモリ使用量を比較

- 行列ベクトル積: 密行列データ型の約 $\frac{1}{10}$ の実行時間
- CG法: 実行時間は $\frac{1}{18} \sim \frac{1}{35}$, 使用メモリは $\frac{1}{17} \sim \frac{1}{45}$



MuPATについて

- Webで公開(個人ホームページ, ATOMS)
 - <http://www.mi.kagu.tus.ac.jp/qupat.html>
 - http://atoms.scilab.org/toolboxes/dd_qd (2011/9/29~)
- 最新バージョン: ver 0.2 (2012/3/12)
 - MuPAT (OS依存無し)
 - MuPAT_win (C利用, Windows向け)
 - MuPAT_maclin (C利用, Mac, Linux向け)
- ダウンロード数(ここ数ヶ月で増加)
 - ATOMS で累計 508 ダウンロード (2012/7/10 時点)
 - Ver 0.2 のみで 437 ダウンロード
- 疎行列は今後公開します