**2021-02-24**

**国防科技大学计算机研究所**

**用户手册**

**并行代数多重网格算法库**

**YHAMG**

目录

[一 软件概述 1](#_Toc103238892)

[1.1 软件功能 1](#_Toc103238893)

[1.2 软件结构 1](#_Toc103238894)

[1.3 软件特点 1](#_Toc103238895)

[二 软件安装 2](#_Toc103238896)

[2.1 系统配置要求 2](#_Toc103238897)

[2.2 安装YHAMG 2](#_Toc103238898)

[2.3 链接YHAMG 2](#_Toc103238899)

[三 应用程序接口 3](#_Toc103238900)

[3.1 向量 5](#_Toc103238901)

[3.1.1 struct Vector 5](#_Toc103238902)

[3.1.2 struct MultiVector 6](#_Toc103238903)

[3.1.3 struct ParVector 7](#_Toc103238904)

[3.1.4 struct ParMultiVector 8](#_Toc103238905)

[3.2 矩阵/算子 9](#_Toc103238906)

[3.2.1 class ParOperator 9](#_Toc103238907)

[3.2.2 class ParOperatorCompound 9](#_Toc103238908)

[3.2.3 struct CSRMatrix 10](#_Toc103238909)

[3.2.4 struct ParCSRMatrix 11](#_Toc103238910)

[3.3 线性求解器 12](#_Toc103238911)

[3.3.1 class ParSolverCG 12](#_Toc103238912)

[3.3.2 class ParSolverGMRES 13](#_Toc103238913)

[3.3.3 class ParSolverBCGS 14](#_Toc103238914)

[3.4 特征求解器 15](#_Toc103238915)

[3.4.1 class ParEigenSolverLOBPCG 15](#_Toc103238916)

[3.5 预条件子接口 16](#_Toc103238917)

[3.5.1 class ParCSRPrecondJacobi 16](#_Toc103238918)

[3.5.2 class ParCSRPrecondSOR 17](#_Toc103238919)

[3.5.3 class ParCSRPrecondILU0 18](#_Toc103238920)

[3.5.4 class ParCSRPrecondAMG 19](#_Toc103238921)

[3.6 矩阵构造工具 21](#_Toc103238922)

[3.6.1 class ParCSRCreator 21](#_Toc103238923)

[3.6.2 class ParCSRAccessor 21](#_Toc103238924)

[3.6.3 class ParCSRMutator 22](#_Toc103238925)

[3.7 矩阵生成器 23](#_Toc103238926)

[3.7.1 class ParCSRGenerator7pt 23](#_Toc103238927)

[3.7.2 class ParCSRGenerator27pt 24](#_Toc103238928)

[3.8 线性代数 25](#_Toc103238929)

[3.8.1 function ParVecAXPBY 25](#_Toc103238930)

[3.8.2 function ParVecAXPBYPCZ 25](#_Toc103238931)

[3.8.3 function ParVecDot 26](#_Toc103238932)

[3.8.4 function ParVecElemMul 26](#_Toc103238933)

[3.8.5 function ParVecRecip 27](#_Toc103238934)

[3.8.6 function ParCSRMatVec 27](#_Toc103238935)

[3.8.7 function ParCSRMatAdd 28](#_Toc103238936)

[3.8.8 function ParCSRMatMul 28](#_Toc103238937)

[3.8.9 function ParCSRTrans 29](#_Toc103238938)

[3.8.10 function ParCSRScale 29](#_Toc103238939)

[3.8.11 function ParCSRScaleRows 30](#_Toc103238940)

[3.8.12 function ParCSRScaleCols 30](#_Toc103238941)

[3.8.13 function ParCSRDiagonal 31](#_Toc103238942)

[3.8.15 function ParCSREliminZeros 31](#_Toc103238943)

[四 示例 32](#_Toc103238944)

[4.1 运行 32](#_Toc103238945)

[4.2 参数列表 33](#_Toc103238946)

# 一 软件概述

## 1.1 软件功能

YHAMG是由国防科技大学研发的并行代数多重网格算法库，主要用于求解形如Ax = b的稀疏线性系统，其中A是一个分布式大型稀疏矩阵，x, b是实向量。YHAMG的目的是在大规模并行计算机上为用户提供高性能可扩展的并行线性求解器。求解方案是预条件子和子空间迭代法，包含了以下算法：

* 稀疏线性代数基本运算；
* 子空间迭代法，包括CG、GMRES和BiCGStab；
* 经典代数多重网格算法（AMG）；
* Jacobi、SOR、SSOR基本迭代法；
* 不完全LU分解。

## 1.2 软件结构

YHAMG主要包括以下模块：

* 线性代数：提供向量，矩阵/算子和线性代数运算；
* 线性求解器：提供并行子空间迭代法；
* 预条件子：提供代数预条件子。

## 1.3 软件特点

YHAMG是用C++编写的，具有以下特点：

* 面向对象设计，支持MPI和OpenMP；
* 完整独立的代码，不依赖其它库；
* 良好的算法可扩展性和并行可扩展性，适用于大型稀疏线性系统求解。

# 二 软件安装

## 2.1 系统配置要求

**硬件要求：**

内存4G以上，CPU主频2G以上。

**软件环境要求：**

* 操作系统：Ubuntu，版本16.04；
* MPI编译子：MPICH，版本3.2.1；
* C++编译子：GCC，版本7.1。

## 2.2 安装YHAMG

1. 解压YHAMG包到安装目录。
2. 执行make。

## 2.3 链接YHAMG

用户需要添加下面的编译命令来链接YHAMG库。

* 添加库路径： -L$YHAMG\_DIR/lib；
* 添加库链接： -lyhamg；
* 添加头文件路径： -I$YHAMG\_DIR/include；

其中$YHAMG\_DIR表示YHAMG主目录的路径。并且在代码的标头中包含：

#include <yhamg.h>

# 三 应用程序接口

YHAMG提供的应用程序接口包括向量、矩阵/算子、预条件子、求解器和线性代数函数等，如下表所示。

|  |  |
| --- | --- |
| **名称** | **描述** |
| **struct** Vector | **向量** |
| struct MultiVector | **多维向量** |
| **struct ParVector** | **并行向量** |
| struct ParMultiVector | **并行多维向量** |
| class Operator | **算子** |
| * struct COOMatrix | **COO矩阵** |
| * struct CSRMatrix | **CSR矩阵** |
| * struct BSRMatrix | **BSR矩阵** |
| class ParOp | **并行算子** |
| * class ParOperatorZero | **零算子** |
| * class ParOperatorIdent | **恒等算子** |
| * class ParOperatorCompound | **复合算子** |
| * class ParOperatorPoly | **算子**多项式 |
| * struct ParCSRMatrix | **并行CSR矩阵** |
| * class ParCSRPrecond | **并行CSR**预条件子 |
| - class ParCSRPrecondJacobi | Jacobi预条件子 |
| - class ParCSRPrecondSOR | SOR预条件子 |
| - class ParCSRPrecondILU0 | ILU(0)预条件子 |
| - class ParCSRPrecondBJSOR | Block Jacobi SOR预条件子 |
| - class ParCSRPrecondBJILU | Block Jacobi ILU预条件子 |
| - class ParCSRPrecondAMG | AMG预条件子 |
| * struct ParBSRMatrix | **并行BSR矩阵** |
| * class ParBSRPrecond | **并行BSR**预条件子 |
| - class ParBSRPrecondJacobi | Jacobi预条件子 |
| - class ParBSRPrecondSOR | SOR预条件子 |
| - class ParBSRPrecondILU0 | ILU(0)预条件子 |
| - class ParBSRPrecondBJSOR | Block Jacobi SOR预条件子 |
| - class ParBSRPrecondBJILU | Block Jacobi ILU预条件子 |
| - class ParBSRPrecondAMG | AMG预条件子 |
| class ParCSRCreator | **并行**CSR创建器 |
| class ParCSRAccessor | **并行**CSR访问器 |
| class ParCSRMutator | **并行**CSR更改器 |
| class ParBSRCreator | **并行**BSR创建器 |
| class ParBSRAccessor | **并行**BSR访问器 |
| class ParBSRMutator | **并行**BSR更改器 |
| class ParSolver | 并行求解器 |
| * class ParSolverCG | 共轭梯度求解器 |
| * class ParSolverGMRES | 广义最小残差求解器 |
| * class ParSolverBCGS | 稳定双共轭梯度求解器 |
| * class ParSolverPipeCG | Piplined CG求解器 |
| * class ParSolverPipeGMRES | Piplined GMRES求解器 |
| * class ParSolverPipeBCGS | Piplined BiCGStab求解器 |
| class ParEigenSolver | 并行特征求解器 |
| * class ParEigenSolverLOBPCG | LOBPCG 特征求解器 |
| class ParCSRGenerator | 并行CSR矩阵生成器 |
| * class ParCSRGenerator7pt | Laplacian矩阵生成器 |
| * class ParCSRGenerator27pt | 27点矩阵生成器 |
| class ParBSRGenerator | 并行BSR矩阵生成器 |
| * class ParBSRGenerator7pt | Laplacian矩阵生成器 |
| * class ParBSRGenerator27pt | 27点矩阵生成器 |
| function ParVecAXPBY | 计算 b = a\*x + b\*y。 |
| function ParVecAXPBYPCZ | 计算 z = a\*x + b\*y + c\*z。 |
| function ParVecDot | 计算向量内积xT\*y。 |
| function ParVecElemMul | 计算向量逐元素乘y = x.\*y。 |
| function ParVecRecip | 计算向量倒数x = 1./x。 |
| function ParCSRMatVec | 计算矩阵向量乘y = A\*x。 |
| function ParCSRMatAdd | 计算矩阵加法C = A + B。 |
| function ParCSRMatMul | 计算矩阵乘法C = A\*B。 |
| function ParCSRTrans | 计算矩阵转置B = AT。 |
| function ParCSRScale | 计算 A = a\*A。 |
| function ParCSRScaleRows | 计算 A = diag(x)\*A。 |
| function ParCSRScaleCols | 计算 A = A \* diag(x)。 |
| function ParCSRDiag | 矩阵对角线/生成对角矩阵。 |
| function ParCSREliminZeros | 消除矩阵零非零元素。 |
| function ParBSRMatVec | 计算矩阵向量乘y = A\*x。 |
| function ParBSRMatAdd | 计算矩阵加法C = A + B。 |
| function ParBSRMatMul | 计算矩阵乘法C = A\*B。 |
| function ParBSRTrans | 计算矩阵转置B = AT。 |
| function ParBSRScale | 计算 A = a\*A。 |
| function ParBSRScaleRows | 计算 A = diag(x)\*A。 |
| function ParBSRScaleCols | 计算 A = A \* diag(x)。 |
| function ParBSRDiag | 矩阵对角线。 |
| function ParBSREliminZeros | 消除矩阵零非零元素。 |

## 3.1 向量

### 3.1.1 struct Vector

**声明**

#include <yhamg.h>

struct Vector;

**描述**

向量。

**成员**

int size;

向量长度。

double\* values;

向量的值。

**成员函数**

void Resize(int n);

设置向量长度并分配内存。

void Free();

释放内存空间并初始化。

void Reference(Vector& x);

设为x的引用。

void Fill(double a);

用标量a填充向量。

void FillRandom();

用(0, 1)之间的随机数填充向量。

void Copy(Vector& x);

复制向量x的值。

void Scale(double a);

乘以标量a。

void AddScaled(double a, Vector& x);

加a\*x。

void Add2Scaled(double a, Vector& x, double b, Vector& y);

加a\*x + b\*y。

**位置**

src/vec/Vector.cpp

### 3.1.2 struct MultiVector

**声明**

#include <yhamg.h>

struct MultiVector;

**描述**

多维向量。

**成员**

int size;

向量长度。

int num\_vectors;

向量数目。

double\* values;

向量的值。

**成员函数**

void Allocate(int n, int m);

分配m个长度为n的向量。

void Free();

释放内存空间并初始化。

Vector& operator()(int j);

返回第j个向量的引用。

void Reference(MultiVector& X);

设为X的引用。

**位置**

src/vec/MultiVector.cpp

### 3.1.3 struct ParVector

**声明**

#include <yhamg.h>

struct ParVector;

**描述**

并行向量。

**成员**

MPI\_Comm comm;

MPI通信子。

Vector local;

Local向量。

**成员函数**

void Resize(int n);

设置向量长度并分配内存。

void Free();

释放内存空间并初始化。

void Reference(Vector& x);

设为x的引用。

void Fill(double a);

用标量a填充向量。

void FillRandom();

用(0, 1)之间的随机数填充向量。

void Copy(Vector& x);

复制向量x。

void Scale(double a);

乘以标量a。

void AddScaled(double a, Vector& x);

加a\*x。

void Add2Scaled(double a, Vector& x, double b, Vector& y);

加a\*x + b\*y。

**位置**

src/vec/ParVector.cpp

### 3.1.4 struct ParMultiVector

**声明**

#include <yhamg.h>

struct ParMultiVector;

**描述**

并行多维向量。

**成员**

MPI\_Comm comm;

MPI通信子

MultiVector local;

Local多维向量

**成员函数**

void Allocate(int n, int m);

分配m个长度为n的向量。

void Free();

释放内存空间并初始化。

ParVector& operator()(int j);

返回第j个向量的引用。

void Reference(ParMultiVector& X);

设为X的引用。

**位置**

src/vec/ParMultiVector.cpp

## 3.2 矩阵/算子

### 3.2.1 class ParOperator

**声明**

#include <yhamg.h>

class ParOperator;

**描述**

并行算子。

**成员函数**

int OutSize();

输出向量长度。

int InSize();

输入向量长度。

void Apply(ParVector& x, ParVector& y);

应用算子。

**位置**

src/ operator /ParOperator.cpp

### 3.2.2 class ParOperatorCompound

**声明**

#include <yhamg.h>

class ParOperatorCompound;

**描述**

复合算子alpha\*A\*B + beta\*C + gamma。

**成员函数**

ParCompoundOperator(double beta, ParOperator& C, double gamma);

ParCompoundOperator(double alpha, ParOperator& B, double beta, ParOperator& C,

double gamma);

ParCompoundOperator(double alpha, ParOperator& A, ParOperator& B, double gamma);

ParCompoundOperator(double alpha, ParOperator& A, ParOperator& B, double beta,

ParOperator& C, double gamma);

构造复合算子。

**位置**

src/operator/ParOperatorCompound.cpp

### 3.2.3 struct CSRMatrix

**声明**

#include <yhamg.h>

struct CSRMatrix;

**描述**

CSR矩阵。

**成员**

int size[2];

矩阵的行数和列数。

int\* rowptr;

非零元素的行偏移。

int\* colind;

非零元素的列索引。

double\* values;

非零元素的值。

**成员函数**

void Free();

释放内存空间并初始化。

void Reference(CSRMatrix& A);

设为A的引用。

**位置**

src/csr/CSRMatrix.cpp

### 3.2.4 struct ParCSRMatrix

**声明**

#include <yhamg.h>

struct ParCSRMatrix;

**描述**

并行CSR矩阵。

**成员**

MPI\_Comm comm;

MPI通信子。

CSRMatrix local;

Local矩阵块。

CSRMatrix exter;

External矩阵块。

int nnbr;

邻居进程数目。

int\* nbrrank;

邻居进程rank。

int\* recvptr;

外点偏移。

int\* recvind;

外点局部索引。

**成员函数**

void Free();

释放内存空间并初始化。

void SetupHalo();

构造通信环。

void Reference(ParCSRMatrix& A) ;

设为A的引用。

**位置**

src/csr/ParCSRMatrix.cpp

## 3.3 线性求解器

### 3.3.1 class ParSolverCG

**声明**

#include <yhamg.h>

class ParSolverCG;

**描述**

共轭梯度法求解器。

**成员**

int MaxIters;

最大迭代次数。默认值为500。

double Tolerance;

停止误差。默认值为1.0e-08。当norm(b – A\*x) / norm(b) < Tolerance，迭代停止。

bool PrintStats;

打印迭代，残差范数等统计信息。默认值为0。

**成员函数**

ParSolverCG (ParOperator& A);

ParSolverCG (ParOperator& A, ParOperator& P);

构造线性求解器（A：系数矩阵，P：预条件子）。

void operator()(ParVector& b, ParVector& x, int& iter, double& relres);

求解线性系统A\*x = b。

**位置**

src/solver/ParSolverCG.cpp

### 3.3.2 class ParSolverGMRES

**声明**

#include <yhamg.h>

class ParSolverGMRES;

**描述**

广义最小残差法求解器。

**成员**

int Restart;

GMRES重启的迭代次数。默认值为10。

int MaxIters;

最大迭代次数。默认值为500。

double Tolerance;

停止误差。默认值为1.0e-08。当norm(b – A\*x) / norm(b) < Tolerance，迭代停止。

bool PrintStats;

打印迭代，残差范数等统计信息。默认值为0。

**成员函数**

ParSolverGMRES (ParOperator& A);

ParSolverGMRES (ParOperator& A, ParOperator& P);

构造线性求解器（A：系数矩阵，P：预条件子）。

void operator()(ParVector& b, ParVector& x, int& iter, double& relres);

求解线性系统A\*x = b。

**位置**

src/solver/ParSolverGMRES.cpp

### 3.3.3 class ParSolverBCGS

**声明**

#include <yhamg.h>

class ParSolverBCGS;

**描述**

稳定双共轭梯度法求解器。

**成员**

int MaxIters;

最大迭代次数。默认值为500。

double Tolerance;

停止误差。默认值为1.0e-08。当norm(b – A\*x) / norm(b) < Tolerance，迭代停止。

bool PrintStats;

打印迭代，残差范数等统计信息。默认值为0。

**成员函数**

ParSolverBCGS (ParOperator& A);

ParSolverBCGS (ParOperator& A, ParOperator& P);

构造线性求解器（A：系数矩阵，P：预条件子）。

void operator()(ParVector& b, ParVector& x, int& iter, double& relres);

求解线性系统A\*x = b。

**位置**

src/solver/ParSolverBCGS.cpp

## 3.4 特征求解器

### 3.4.1 class ParEigenSolverLOBPCG

**声明**

#include <yhamg.h>

class ParEigenSolverLOBPCG;

**描述**

LOBPCG特征求解器。

**成员**

int MaxIters;

最大迭代次数。默认值为500。

double Tolerance;

停止误差。默认值为1.0e-08。

bool PrintStats;

打印迭代，残差范数等统计信息。默认值为0。

**成员函数**

ParEigenSolverLOBPCG (ParOperator& A);

ParEigenSolverLOBPCG (ParOperator& A, ParMultiVector& Y);

ParEigenSolverLOBPCG (ParOperator& A, ParOperator& T);

ParEigenSolverLOBPCG (ParOperator& A, ParOperator& T, ParMultiVector& Y);

ParEigenSolverLOBPCG (ParOperator& A, ParOperator& B, ParOperator& T);

ParEigenSolverLOBPCG (ParOperator& A, ParOperator& B, ParOperator& T,

ParMultiVector& Y);

构造特征求解器（A, B：系数矩阵，T：预条件子，Y：约束）。

void operator()(ParVector& x, double& lambda, int& iter, double& res);

void operator()(ParMultiVector& X, Vector& Lambda, int& iter, Vector& Res);

求解广义特征问题A\*x = lambda\*B\*x。

**位置**

src/eigen/ParEigenSolverLOBPCG.cpp

## 3.5 预条件子接口

### 3.5.1 class ParCSRPrecondJacobi

**声明**

#include <yhamg.h>

class ParCSRPrecondJacobi;

**描述**

Jacobi预条件子。

**成员函数**

void Setup(ParCSRMatrix& A);

构造预条件子。

void Free();

销毁预条件子并初始化。

**位置**

src/csr/ParCSRPrecondJacobi.cpp

### 3.5.2 class ParCSRPrecondSOR

**声明**

#include <yhamg.h>

class ParCSRPrecondSOR;

**描述**

SOR预条件子。

**成员**

int RelaxationType;

松弛类型。0：Forward SOR, 1: Backward SOR, 2（默认）：Symmetric SOR。

double RelaxationFactor;

松弛因子。

**成员函数**

void Setup(ParCSRMatrix& A);

构造预条件子。

void Free();

销毁预条件子并初始化。

**位置**

src/csr/ParCSRPrecondSOR.cpp

### 3.5.3 class ParCSRPrecondILU0

**声明**

#include <yhamg.h>

class ParCSRPrecondILU0;

**描述**

ILU(0)预条件子。

**成员函数**

void Setup(ParCSRMatrix& A);

构造预条件子。

void Free();

销毁预条件子并初始化。

**位置**

src/csr/ParCSRPrecondILU0.cpp

### 3.5.4 class ParCSRPrecondAMG

**声明**

#include <yhamg.h>

class ParCSRPrecondAMG;

**描述**

经典代数多重网格预条件子。

**成员**

int MaxLevels;

最大网格层数。默认值为20。

int CoarseSize;

最粗网格层规模。默认值为8。当网格层未知量少于CoarseSize时，将该层作为最粗网格层，并停止生成新的网格层。

double StrengthThreshold;

强度阈值。默认值为0.25。

int AggNumLevels;

使用Aggressive粗化的网格层数。默认值为0。对于大规模三维问题，通常建议使用1层Aggressive粗化来降低复杂度。

int CoarsenType;

指定粗化算法。0（默认）：HMIS，1：PMIS。

int InterpType;

指定插值算法。0（默认）：Long Range Interpolation，1：Smoothed Aggregation。

int InterpMinElements;

最小插值数目。默认值为4。

int InterpMaxElements;

最大插值数目。默认值为4。

double TruncationFactor;

插值截断因子。默认值为0.1。

double SparsificationThreshold;

粗网格矩阵稀疏化阈值。默认值为0.01。稀疏化阈值建议选择范围为0.0~0.1。

int CycleType;

指定循环类型。0（默认）：V循环，1：W循环，2：F循环。

int PreSweeps;

粗网格校正前的松弛迭代次数。默认值为1。

int PostSweeps;

粗网格校正后的松弛迭代次数。默认值为1。

int CoarseSweeps;

求解最粗网格层的松弛迭代次数。默认值为2。

int SmoothType;

指定光滑类型。0：Jacobi，1（默认）：SOR，2：ILU，3：Chebyshev。

double JacobiFactor;

Jacobi松弛因子。默认值为0.75。

int SORType;

SOR类型。0：Forward SOR，1：BackwardSOR，2（默认）：Symmetric SOR。

double SORFactor;

SOR松弛因子。默认值为1.0。

int ILUMaxFillins;

ILU最大填充数目。默认值为0。

double ILUDropTolerance;

ILU调降容差。默认值为0.01。

int ChebyshevOrder;

Chebyshev多项式阶数。默认值为4。

double ChebyshevLower;

Chebyshev下界。默认值为0.3。

double ChebyshevUpper;

Chebyshev上界。默认值为1.1。

bool PrintStats;

打印网格层次结构，网格复杂度等统计信息。默认值为0。

**成员函数**

void Setup(ParCSRMatrix& A);

构造预条件子。

void Free();

销毁预条件子并初始化。

**位置**

src/csr/ParCSRPrecondAMG.cpp

## 3.6 矩阵构造工具

### 3.6.1 class ParCSRCreator

**声明**

#include <yhamg.h>

class ParCSRCreator;

**描述**

并行CSR创建器。

**成员函数**

ParCSRCreator(MPI\_Comm comm, int n, int m);

构造并行CSR创建器。

void operator()(ParCSRMatrix& A);

生成一个全零矩阵A。

**位置**

src/csr/ParCSRCreator.cpp

### 3.6.2 class ParCSRAccessor

**声明**

#include <yhamg.h>

class ParCSRAccessor;

**描述**

并行CSR访问器。

**成员函数**

ParCSRAccessor (ParCSRMatrix& A);

构造并行CSR访问器。

double\* Find(int row, global col);

返回对应(row, col)位置的非零元素的值的地址；若缺少该非零元素，返回0。全局索引global类型包含局部索引indl和所属进程from，其相关的构造函数：

global(int indl, int from);

**位置**

src/csr/ParCSRAccessor.cpp

### 3.6.3 class ParCSRMutator

**声明**

#include <yhamg.h>

class ParCSRMutator;

**描述**

并行CSR更改器。

**成员函数**

ParCSRMutator (ParCSRMatrix& A);

构造并行CSR更改器。

double\* Find(int row, global col);

返回对应(row, col)位置的非零元素的值的地址；若缺少该非零元素，返回0。

double\* Insert(int row, global col);

插入对应(row, col)位置的非零元素，并返回该非零元素的值的地址。

void Erase(int row, global col);

擦除对应(row, col)位置的非零元素。

void operator()(ParCSRMatrix& B);

应用所有更改并生成一个新的矩阵B（更改不会应用到矩阵A）。

**位置**

src/csr/ParCSRMutator.cpp

## 3.7 矩阵生成器

### 3.7.1 class ParCSRGenerator7pt

**声明**

#include <yhamg.h>

class ParCSRGenerator7pt;

**描述**

Laplacian矩阵生成器。

**成员**

MPI\_Comm comm;

MPI通信子。

int nx, ny, nz;

本地规模。

int Px, Py, Pz;

进程拓扑。

**成员函数**

void operator()(ParCSRMatrix& A) ;

生成矩阵A。

**位置**

src/csr/ParCSRGenerator7pt.cpp

### 3.7.2 class ParCSRGenerator27pt

**声明**

#include <yhamg.h>

class ParCSRGenerator27pt;

**描述**

Laplacian 27点矩阵生成器。

**成员**

MPI\_Comm comm;

MPI通信子。

int nx, ny, nz;

本地规模。

int Px, Py, Pz;

进程拓扑。

**成员函数**

void operator()(ParCSRMatrix& A) ;

生成矩阵A。

**位置**

src/csr/ParCSRGenerator27pt.cpp

## 3.8 线性代数

### 3.8.1 function ParVecAXPBY

**声明**

#include <yhamg.h>

void ParVecAXPBY(double a, ParVector& x, double b, ParVector& y);

**描述**

计算 y = a\*x + b\*y。

**输入参数**

a, b - 标量

x, y - 向量

**输出参数**

y- 向量

**位置**

src/vec/ParVector.cpp

### 3.8.2 function ParVecAXPBYPCZ

**声明**

#include <yhamg.h>

void ParVecAXPBYPCZ (double a, ParVector& x, double b, ParVector& y, double c,

ParVector& z);

**描述**

计算 z = a\*x + b\*y + c\*z。

**输入参数**

a, b, c - 标量

x, y, z - 向量

**输出参数**

z- 向量

**位置**

src/vec/ParVector.cpp

### 3.8.3 function ParVecDot

**声明**

#include <yhamg.h>

double ParVecDot(ParVector& x, ParVector& y);

**描述**

计算向量内积xT\*y。

**输入参数**

x, y - 向量

**返回值**

ret- 内积

**位置**

src/vec/ParVector.cpp

### 3.8.4 function ParVecElemMul

**声明**

#include <yhamg.h>

void ParVecElemMul (ParVector& x, ParVector& y);

**描述**

计算向量逐元素乘y = x.\*y。

**输入参数**

x, y - 向量

**输出参数**

y - 向量

**位置**

src/vec/ParVector.cpp

### 3.8.5 function ParVecRecip

**声明**

#include <yhamg.h>

double ParVecRecip (ParVector& x);

**描述**

计算x = 1./x。

**输入参数**

x - 向量

**输出参数**

x - 向量

**位置**

src/vec/ParVector.cpp

### 3.8.6 function ParCSRMatVec

**声明**

#include <yhamg.h>

void ParCSRMatVec(ParCSRMatrix& A, ParVector& x, ParVector& y);

**描述**

计算矩阵向量乘y = A\*x。

**输入参数**

x - 向量

A - 稀疏矩阵

**输出参数**

**y** - 向量

**位置**

src/csr/ParCSRMatrix.cpp

### 3.8.7 function ParCSRMatAdd

**声明**

#include <yhamg.h>

void ParCSRMatAdd(ParCSRMatrix& A, ParCSRMatrix& B, ParCSRMatrix& C);

**描述**

计算矩阵加法C = A + B。

**输入参数**

A, B - 稀疏矩阵

**输出参数**

C - 稀疏矩阵

**位置**

src/csr/ParCSRMatrix.cpp

### 3.8.8 function ParCSRMatMul

**声明**

#include <yhamg.h>

void ParCSRMatMul(ParCSRMatrix& A, ParCSRMatrix& B, ParCSRMatrix& C);

**描述**

计算矩阵乘法C = A\*B。

**输入参数**

A, B - 稀疏矩阵

**输出参数**

C - 稀疏矩阵

**位置**

src/csr/ParCSRMatrix.cpp

### 3.8.9 function ParCSRTrans

**声明**

#include <yhamg.h>

void ParCSRTrans (ParCSRMatrix& A, ParCSRMatrix& B);

**描述**

计算矩阵转置B = AT。

**输入参数**

A - 稀疏矩阵

**输出参数**

B - 稀疏矩阵

**位置**

src/csr/ParCSRMatrix.cpp

### 3.8.10 function ParCSRScale

**声明**

#include <yhamg.h>

void ParCSRScale(double a, ParCSRMatrix& A);

**描述**

计算A = a \* A。

**输入参数**

a - 标量

A - 稀疏矩阵

**输出参数**

A - 稀疏矩阵

**位置**

src/csr/ParCSRMatrix.cpp

### 3.8.11 function ParCSRScaleRows

**声明**

#include <yhamg.h>

void ParCSRScaleRows(ParVector& x, ParCSRMatrix& A);

**描述**

计算A = diag(x)\*A。

**输入参数**

x - 向量

A - 稀疏矩阵

**输出参数**

A - 稀疏矩阵

**位置**

src/csr/ParCSRMatrix.cpp

### 3.8.12 function ParCSRScaleCols

**声明**

#include <yhamg.h>

void ParCSRScaleCols(double a, ParCSRMatrix& A);

**描述**

计算A = A\* diag(x)。

**输入参数**

x - 向量

A - 稀疏矩阵

**输出参数**

A - 稀疏矩阵

**位置**

src/csr/ParCSRMatrix.cpp

### 3.8.13 function ParCSRDiagonal

**声明**

#include <yhamg.h>

void ParCSRDiagonal(ParCSRMatrix& A, ParVector& D);

**描述**

获取矩阵对角线。

**输入参数**

A - 稀疏矩阵

**输出参数**

D - 矩阵对角线

**位置**

src/csr/ParCSRMatrix.cpp

### 3.8.15 function ParCSREliminZeros

**声明**

#include <yhamg.h>

void ParCSREliminZeros (ParCSRMatrix& A);

**描述**

消除矩阵零非零元。

**输入参数**

A - 稀疏矩阵

**输出参数**

A - 稀疏矩阵

**位置**

src/csr/ParCSRMatrix.cpp

# 四 示例

YHAMG提供了一个示例程序，于test目录下。运行前，请设置系统环境变量OMP\_NUM\_THREADS控制OpenMP线程数。

export OMP\_NUM\_THREADS=1

## 4.1 运行

> mpirun -n 8 ./test -laplace -n 100 100 100 -P 2 2 2 -pre 4 -solver 0

# Test Problem

--------------------------------------------------

3D Laplace Problem on a Cube

Global Problem Size: (200, 200, 200)

Local Domain Size: (100, 100, 100)

MPI Process Size: (2, 2, 2)

# AMG Setup Phase

--------------------------------------------------

Multigrid Hierarchy:

Level Rows Nz

--------------------------------------------------

0 8000000 55760000

1 482611 18098598

2 72128 4481012

3 9449 670789

4 936 45482

5 130 4268

6 19 196

7 4 15

--------------------------------------------------

Number of Levels: 8

Grid Complexity: 1.07066

Operator Complexity: 1.41787

Setup Time: 2.76183

# CG Solve Phase

--------------------------------------------------

Iter Residual

--------------------------------------------------

0 2828.43

1 16795.2

2 2702.6

3 491.845

4 89.0701

5 14.9227

6 2.49919

7 0.417837

8 0.0692848

9 0.0115655

10 0.00192553

11 0.000324108

12 5.44729e-05

13 9.07726e-06

--------------------------------------------------

Iterations: 13

Final Relative Residual: 3.2093e-09

Solve Time: 1.91141

Time/Iteration: 0.147032

## 4.2 参数列表

-laplace 3D Laplace problem on a cube(default)

-27pt 3D Laplace problem with 27-point stencil

-n <nx ny nz> local domain size

-b <bx by bz> nonzero block size

-P <Px Py Pz> MPI process size

-pre <ID> preconditioner ID

0(default): Jacobi

1: SOR

2: ILU(0)

3: Chebyshev

4: AMG

-solver <ID> solver ID

0(default): CG

1: GMRES(10)

2: BiCGStab