Vefemz: A C++ program package for VEM and FEM

Jikun Zhao
Zhengzhou University

Contents

1 Introduction

Vefemz is a C++ program package to implement the computations for the virtual element method (VEM) and finite element method (FEM). This package can be found at *https://github.com/jkzhaozzu/Vefemz*. For linear algebra such as matrices, vectors and numerical solvers, Vefemz uses the C++ template library *Eigen*, which can be found at *http://eigen.tuxfamily.org/index.php?title=Main_Page*. Vefemz is designed for VEM at the beginning, and then extended to FEM. It consists of several types of classes as follows:

1. polynomial space class
2. numerical quadrature class
3. mesh class
4. degree of freedom class
5. virtual element class
6. problem class
7. model class for VEM
8. finite element class
9. model class for FEM

2 Polynomial space class

In the header file *polynomialspace.h*, for the polynomial space there is a base class *PolynomialSpace* with one public member

int p : the order of polynomial space;

and four public virtual member functions

1. *virtual double GetValue(int i,double x,double y,double hE,double xE,double yE)* to return the value of the i-th basis function at (x, y) in $\mathbb{P}_p$, where hE is mesh element diameter and (xE, yE) is the barycenter of mesh element;
2. *virtual double GetDerivative(int i,int dxm,int dyn,double x,double y,double hE,double xE,double yE)* to return the (dxm, dyn)-th derivative value of the i-th basis function at (x, y) in $\mathbb{P}_p$;
3. *virtual void GetValue(int i,double x,double y,double hE,double xE,double yE,double *value)* to get the value of the i-th basis function at (x, y) in $\mathbb{P}_p^2$, which is saved at the array $value$;
4. *virtual void GetDerivative(int i,int dxm,int dyn,double x,double y,double hE,double xE,double yE,double *value)* to get the (dxm, dyn)-th derivative value of the i-th basis function at (x, y) in $\mathbb{P}_p^2$, which is saved at the array $value$.

The base class *PolynomialSpace* is inherited by the following polynomial space classes:

1. *ScaledMonomialSpace* class for the scalar-valued polynomial space $\mathbb{P}_p$ of order p or less;
2. *ScaledMonomialSpaceV* class for the vector-valued polynomial space $\mathbb{P}_p^2$ of order p or less;
3. *GradPolynomialSpace* class for the gradient $\mathbb{G}_p$ of polynomial space $\mathbb{P}_{p+1}$;
4. *GradPolynomialOrthogonalSpace* class for the polynomial subspace $\mathbb{G}_p^\perp$ orthogonal to $\mathbb{G}_p$ in $\mathbb{P}_p^2$.

3 Numerical quadrature class

In the header file *quadrature.h*, there are five numerical quadrature class as follows:

1. *GaussLegendreQuadrature* class
2. *GaussLobattoQuadrature* class
3. *GaussLobattoTypeQuadrature* class
4. *TriangleQuadrature* class
5. *TetrahedronQuadrature* class

4 Mesh class

In the header file *mesh.h*, there is two classes as follows:

1. *Domain* class
2. *PolyMesh* class

5 Degree of freedom class

In the header file *dof.h*, there is a class *DegreeofFreedom*.

6 Virtual element class

6.1 Base class *VirtualElement*

In the header file *virtualelement.h*, there is a base class *VirtualElement* with five public members:

int p: the order of virtual element
PolyMesh &ms
DegreeofFreedom &dof
PolynomialSpace &SMS
TriangleQuadrature &TQ

and many public virtual member functions:

*virtual void GetB_H1(int ElemID,double **B)*
*virtual void GetD(int ElemID,double **D)*
*virtual void GetG_H1(int ElemID,double **G)*

*virtual void GetPistar_H1(int ElemID,double **Pistar)*
*virtual void GetPi_H1(int ElemID,double **Pi)*
*virtual void GetA_H1(int ElemID,double **A)*
*virtual void GetB_H2(int ElemID,double **B)*
*virtual void GetG_H2(int ElemID,double **G)*
*virtual void GetPistar_H2(int ElemID,double **Pistar)*
*virtual void GetPi_H2(int ElemID,double **Pi)*
*virtual void GetA_H2(int ElemID,double **A)*
*virtual void GetB_L2(int ElemID,double **B)*
*virtual void GetG_L2(int ElemID,double **G)*
*virtual void GetPistar_L2(int ElemID,double **Pistar)*
*virtual void GetPi_L2(int ElemID,double **Pi)*
*virtual void GetA_L2(int ElemID,double **A)*
*virtual void GetB_div(int ElemID,double **B)*
*virtual void GetG_div(int ElemID,double **G)*
*virtual void GetPistar_div(int ElemID,double **Pistar)*
*virtual void GetA_div(int ElemID,double **A)*
*virtual double GetIntegralValue(int ElemID,double *X)*
*virtual void GetBdof_BdofVal(FunctionP BFunc,int *Bdof,double *Bdofval)* to get boundary dof ID and corresponding dof value
*virtual void GetDofVal(FunctionP u,double *uI)*
*virtual void GetRHS(int ElemID,FunctionP Source,double *LocF)*
*virtual void GetBdof_BdofVal(FunctionPt BFunc,double t,int *Bdof,double *Bdofval)* to get boundary dof ID and corresponding dof value at t
*virtual void GetDofVal(FunctionPt u,double t,double *uI)*
*virtual void GetRHS(int ElemID,FunctionPt Source,double t,double *LocF)*
*virtual void GetdxB_L2(int ElemID,double **B)*
*virtual void GetdyB_L2(int ElemID,double **B)*
*virtual void GetdxPistar_L2(int ElemID,double **Pistar)*
*virtual void GetdyPistar_L2(int ElemID,double **Pistar)*

The base class *VirtualElement* is inherited by the following virtual element classes:
1. *VEPkC0* class for the H^1-conforming virtual element [2, 3]
2. *VEPkNC* class for the H^1-nonconforming virtual element [1]
3. *VEPkDG* class for the discontinuous piecewise polynomial space
4. *VEPkH2* class for the H^2-conforming virtual element [4]
5. *VEC0PkH2NC* class for the C^0-continuous H^2-nonconforming virtual element [5]
6. *VEMorleyType* class for the Morley-type virtual element [6]
7. *VEPkNCV* class for the vector-valued H^1-nonconforming virtual element with polynomial divergence [7]

6.2 Base class *MixedVirtualElement*

In the header file *mixedvirtualelement.h*, for the mixed virtual element there is a base class *MixedVirtualElement* with two public members:
VirtualElement &VE1 for the first virtual element
VirtualElement &VE2 for the second virtual element
and two public virtual member functions:
*virtual void GetMixedA(int ElemID,double **A)*
*virtual void GetMixedB(int ElemID,double **B)*

The base class *MixedVirtualElement* is inherited by the following virtual element classes:
1. *MVEPkNC* class for the H^1-nonconforming Stokes virtual element consisting of *VEPkNCV* and *VEPkDG* [7]
2. *MVEPkHdivNC* class for the H^1-nonconforming but $H(\mathrm{div})$-conforming Stokes virtual element consisting of *VEPkHdivNCV* and *VEPkDG* [7]

7 VEM function class

In the header file *vemfunction.h*, there is a class *VEMFunction* with a private member:
int size: the number of degree of freedom for the current virtual element
six public member:
PolyMesh &*ms*
DegreeofFreedom &*dof*
PolynomialSpace &*SMS*
TriangleQuadrature &*TQ*
VirtualElement &*VE*
VectorXd pVector: a vector defined by *Eigen::VectorXd* to save the value of d.o.f.
and several public member functions:
VEMFunction(VirtualElement &*ve):SMS(ve.SMS),TQ(ve.TQ),ms(ve.ms),dof(ve.dof),VE(ve), pVector(ve.dof.Dof_Num)*
void Interpolate(FunctionP u) to get u
void Interpolate(FunctionPt u,double t): interpolation at t
const double &*operator[](int n) const* to return the value of n-th d.o.f.
double &*operator[](int n)* to change the value of n-th d.o.f.
VEMFunction &*operator=(const VEMFunction* &*pVer)* to copy pVer
double GetL2ProjValue(double x,double y) to get L2 projection value at (x,y);
double EnergyNorm(SparseMatrix<double> &*A)* to get the norm by X^TAX
double GetAverageValue()

8 Problem class

In the header file *problemmodel.h*, there are some classes for several classical model problems:

1. *PoissonProblem* class
2. *ParabolicProblem* class
3. *ElasticityProblem* class
4. *StokesProblem* class
5. *DarcyStokesProblem* class
6. *NavierStokesProblem* class
7. *BiharmonicProblem* class
8. *PlateContactProblem* class

9 Model class for VEM

In the header file *problemmodel.h*, there are some classes for several models of VEM corresponding to those problem classes:

1. *PoissonModel* class
2. *ParabolicModel* class
3. *ElasticityModel* class
4. *StokesModel* class
5. *DarcyStokesModel* class
6. *NavierStokesModel* class
7. *BiharmonicModel* class
8. *PlateContactModel* class

10 Finite element class

10.1 Base class *finiteelement*

In the header file *finiteelement.h*, there is a base class *FiniteElement* with five public members:
int p: the order of finite element
PolyMesh &*ms*
DegreeofFreedom &*dof*
PolynomialSpace &*SMS*

TriangleQuadrature &*TQ*
and many public virtual member functions:
*virtual void GetD(int ElemID,double **D)*
*virtual void GetBase(int ElemID,double **BF)*
*virtual void GetG_L2(int ElemID,double **G)*
*virtual void GetA_L2(int ElemID,double **A)*
*virtual void GetG_H1(int ElemID,double **G)*
*virtual void GetA_H1(int ElemID,double **A)*
*virtual void GetG_H2(int ElemID,double **G)*
*virtual void GetA_H2(int ElemID,double **A)*
*virtual void GetB_div(int ElemID,double **B)*
*virtual void GetG_div(int ElemID,double **G)*
*virtual void GetPistar_div(int ElemID,double **Pistar)*
*virtual void GetA_div(int ElemID,double **A)*
*virtual double GetIntegralValue(int ElemID,double *X)*
*virtual void GetBdof_BdofVal(FunctionP BFunc,int *Bdof,double *Bdofval)* to get boundary dof ID and corresponding dof value
*virtual void GetDofVal(FunctionP u,double *uI)*
*virtual void GetRHS(int ElemID,FunctionP Source,double *LocF)*
*virtual void GetBdof_BdofVal(FunctionPt BFunc,double t,int *Bdof,double *Bdofval)* to get boundary dof ID and corresponding dof value at t
*virtual void GetDofVal(FunctionPt u,double t,double *uI)*
*virtual void GetRHS(int ElemID,FunctionPt Source,double t,double *LocF)*
*virtual void GetdxB_L2(int ElemID,double **B)*
*virtual void GetdyB_L2(int ElemID,double **B)*
*virtual void GetdxPistar_L2(int ElemID,double **Pistar)*
*virtual void GetdyPistar_L2(int ElemID,double **Pistar)*

The base class *FiniteElement* is inherited by the following finite element classes:
FEPkC0 class for H^1-conforming Lagrange finite element

10.2 Base class *mixedfiniteelement*

In the header file *mixedfiniteelement.h*, there is a class *MFEP1NCRS* with first-order convergence for the stablized mixed FE for linear elasticity on rectangular meshes.

In the future, there will be a basis class *MixedFiniteElement*.......

11 FEM function class

In the header file *femfunction.h*, there is a class *FEMFunction* with a private member:
int size: the number of degree of freedom for the current finite element
six public member:
PolyMesh &*ms*
DegreeofFreedom &*dof*
PolynomialSpace &*SMS*
TriangleQuadrature &*TQ*
VirtualElement &*VE*
VectorXd pVector: a vector defined by *Eigen::VectorXd* to save the value of d.o.f.
and several public member functions:
FEMFunction(FiniteElement &fe):SMS(fe.SMS),TQ(fe.TQ),ms(fe.ms),dof(fe.dof),FE(fe), pVector(fe.dof.Dof_Num)
void Interpolate(FunctionP u) to get u
void Interpolate(FunctionPt u,double t): interpolation at t
const double &operator[](int n) const to return the value of n-th d.o.f.
double &operator[](int n) to change the value of n-th d.o.f.
FEMFunction &operator=(const FEMFunction &pVer) to copy pVer
double GetL2ProjValue(double x,double y) to get L2 projection value at (x,y);
double EnergyNorm(SparseMatrix<double> &A) to get the norm by X^TAX

double GetAverageValue()

12 Model class for FEM

In the header file *FEmodel.h*, there are some classes for several models of FEM corresponding to those problem classes:

1. *FEPoissonModel* class
2. *FEElasticityMixedModel*: A mixed FEM with stabilization for linear elasticity solved by finite element *MFEP1NCRS*

13 VEM examples

In the file folder *example*, there are some VEM examples:
poissonVEPkC0.h
poissonVEPkNC.h
ParabolicVEPkC0.h
ParabolicVEPkNC.h
elasticityVEPkNCV.h
StokesMVEPkNC.h
StokesMVEPkHdivNC.h
DSMVEPkHdivNC.h for the Darcy-Stokes problem
NavierStokesMVEPkNC.h
BiharmVEPkH2.h
BiharmVEC0PkH2NC.h
BiharmVEMorleyType.h
platecontactVEPkH2
platecontactVEC0PkH2NC.h
platecontMorleyType.h

14 FEM examples

In the file folder *FEexample*, there are some FEM examples:
poissonFEPkC0
elasticityMFEP1NCRS.h

References

[1] B. Ayuso de Dios, K. Lipnikov, and G. Manzini, *The nonconforming virtual element method*, ESAIM Math. Model. Numer. Anal., 50 (2016), pp. 879–904.

[2] L. Beirão da Veiga, F. Brezzi, A. Cangiani, G. Manzini, L. D. Marini, and A. Russo, *Basic principles of virtual element methods*, Math. Models Methods Appl. Sci., 23 (2013), pp. 199–214.

[3] L. Beirão da Veiga, F. Brezzi, L. D. Marini, and A. Russo, *The hitchhiker's guide to the virtual element method*, Math. Models Methods Appl. Sci., 24 (2014), pp. 1541–1573.

[4] F. Brezzi and L. D. Marini, *Virtual element methods for plate bending problems*, Comput. Method Appl. Mech. Engrg., 253 (2013), pp. 455–462.

[5] J. Zhao, S. Chen, and B. Zhang, *The nonconforming virtual element method for plate bending problems*, Math. Models Methods Appl. Sci., 26 (2016), pp. 1671–1687.

[6] J. Zhao, B. Zhang, S. Chen, and S. Mao, *The Morley-type virtual element for plate bending problems*, J. Sci. Comput., 76 (2018), pp. 610–629.

[7] J. Zhao, B. Zhang, S. Mao, and S. Chen, *The divergence-free nonconforming virtual element for the Stokes problem*, SIAM J. Numer. Anal., 57 (2019), pp. 2730–2759.