FENNEC

Version 0.0 beta

Generated by Doxygen 1.8.11

Contents

1 Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

- AuxKernel
    - **AccumulatedMaterial** 5
- DGKernel
    - **DGAdvection** 12
        - **DGConcentrationAdvection** 17
        - **DGMomentumAdvection** 32
    - **DGAnisotropicDiffusion** 14
        - **DGMomentumDiffusion** 36
- InitialCondition
    - **ConstantEllipsoidIC** 7
- IntegratedBC
    - **DGFluxBC** 27
        - **DGConcentrationFluxBC** 20
        - **DGMomentumFluxBC** 40
            - **DGContinuumBC** 23
    - **DGFluxLimitedBC** 29
- Kernel
    - **CoupledCoeffTimeDerivative** 9
    - **GAdvection** 45
        - **GConcentrationAdvection** 50
        - **GMomentumAdvection** 53
    - **GAnisotropicDiffusion** 47
        - **GMomentumDiffusion** 57
    - **MomentumAcceleration** 60
    - **MomentumPressureGrad** 64
    - **StressTensor** 66
- MooseApp
    - **fennecApp** 44
- TimeDerivative

2 Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

3 File Index

3.1 File List

Here is a list of all files with brief descriptions:

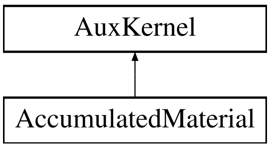
4 Class Documentation

4.1 AccumulatedMaterial Class Reference

AccumulatedMaterial class inherits from AuxKernel.

```
#include <AccumulatedMaterial.h>
```

Inheritance diagram for AccumulatedMaterial:



Public Member Functions

- AccumulatedMaterial (const InputParameters &parameters)

  Standard MOOSE public constructor.

Protected Member Functions

- virtual Real computeValue ()

  Required MOOSE function override.

Protected Attributes

- std::vector< const VariableValue * > _coupled_u

  Pointer list for the non-linear variables.

4.1.1 Detailed Description

AccumulatedMaterial class inherits from AuxKernel.

This class object creates an AuxKernel for use in the MOOSE framework. The AuxKernel will calculate the total accumulation of a non-linear variable that has passed through the elements of the mesh. Thereby creating a running total of all material over time. This is useful for determining the total amount of particles that may be deposited on a surface via settling.

Definition at line 57 of file AccumulatedMaterial.h.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 AccumulatedMaterial::AccumulatedMaterial (const InputParameters & *parameters*)

Standard MOOSE public constructor.

4.1.3 Member Function Documentation

4.1.3.1 virtual Real AccumulatedMaterial::computeValue ( ) [protected], [virtual]

Required MOOSE function override.

This is the function that is called by the MOOSE framework when a calculation of the total system pressure is needed. You are required to override this function for any inherited AuxKernel.

4.1.4 Member Data Documentation

4.1.4.1 std::vector<const VariableValue *> AccumulatedMaterial::_coupled_u [protected]

Pointer list for the non-linear variables.

Definition at line 70 of file AccumulatedMaterial.h.

The documentation for this class was generated from the following file:

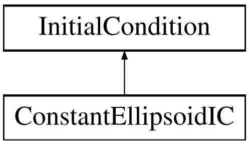
- AccumulatedMaterial.h

4.2 ConstantEllipsoidIC Class Reference

ConcentrationIC class object inherits from InitialCondition object.

```
#include <ConstantEllipsoidIC.h>
```

Inheritance diagram for ConstantEllipsoidIC:



Public Member Functions

- ConstantEllipsoidIC (const InputParameters &parameters)

  Required constructor for objects in MOOSE.
- virtual Real value (const Point &p)

  Required function override for setting the value of the non-linear variable at a given point.

Public Attributes

- Real _x_rad

  Below are parameters to defind the ellipsoid in 3D space.
- Real _y_rad

  Radius of the ellipsoid in y.
- Real _z_rad

  Radius of the ellipsoid in z.
- Real _x_center

  x-coordinate center of the ellipsoid
- Real _y_center

  y-coordinate center of the ellipsoid
- Real _z_center

  z-coordinate center of the ellipsoid
- Real _value_internal

  Value of the non-linear variable inside the ellipsoid.
- Real _value_external

  Value of the non-linear variable outside the ellipsoid.
- Real _smoother_distance

  Distance over which to smooth the non-linear variable from interior to exterior.

4.2.1 Detailed Description

ConcentrationIC class object inherits from InitialCondition object.

This class object inherits from the InitialCondition object in the MOOSE framework. All public and protected members of this class are required function overrides. The object will establish the initial conditions for a species' concentration as constant throughout the domain.

Definition at line 56 of file ConstantEllipsoidIC.h.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 ConstantEllipsoidIC::ConstantEllipsoidIC (const InputParameters & *parameters*)

Required constructor for objects in MOOSE.

4.2.3 Member Function Documentation

4.2.3.1 virtual Real ConstantEllipsoidIC::value (const Point & *p*) `[virtual]`

Required function override for setting the value of the non-linear variable at a given point.

This function passes a point p as an argument. The return value will be the value of the non-linear variable at that point. That information is used to establish the spatially varying initial condition for the given non-linear variable.

4.2.4 Member Data Documentation

4.2.4.1 Real ConstantEllipsoidIC::_smoother_distance

Distance over which to smooth the non-linear variable from interior to exterior.

Definition at line 78 of file ConstantEllipsoidIC.h.

4.2.4.2 Real ConstantEllipsoidIC::_value_external

Value of the non-linear variable outside the ellipsoid.

Definition at line 77 of file ConstantEllipsoidIC.h.

4.2.4.3 Real ConstantEllipsoidIC::_value_internal

Value of the non-linear variable inside the ellipsoid.

Definition at line 76 of file ConstantEllipsoidIC.h.

4.2.4.4 Real ConstantEllipsoidIC::_x_center

x-coordinate center of the ellipsoid

Definition at line 72 of file ConstantEllipsoidIC.h.

4.2.4.5 Real ConstantEllipsoidIC::_x_rad

Below are parameters to defind the ellipsoid in 3D space.

Radius of the ellipsoid in x

Definition at line 69 of file ConstantEllipsoidIC.h.

4.2.4.6 Real ConstantEllipsoidIC::_y_center

y-coordinate center of the ellipsoid

Definition at line 73 of file ConstantEllipsoidIC.h.

4.2.4.7 Real ConstantEllipsoidIC::_y_rad

Radius of the ellipsoid in y.

Definition at line 70 of file ConstantEllipsoidIC.h.

4.2.4.8 Real ConstantEllipsoidIC::_z_center

z-coordinate center of the ellipsoid

Definition at line 74 of file ConstantEllipsoidIC.h.

4.2.4.9 Real ConstantEllipsoidIC::_z_rad

Radius of the ellipsoid in z.

Definition at line 71 of file ConstantEllipsoidIC.h.

The documentation for this class was generated from the following file:

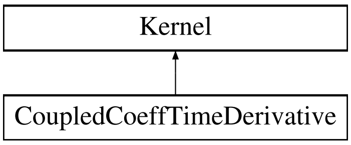
- ConstantEllipsoidIC.h

4.3 CoupledCoeffTimeDerivative Class Reference

CoupledCoeffTimeDerivative class object inherits from Kernel object.

```
#include <CoupledCoeffTimeDerivative.h>
```

Inheritance diagram for CoupledCoeffTimeDerivative:



Public Member Functions

- CoupledCoeffTimeDerivative (const InputParameters &parameters)
  Required constructor for objects in MOOSE.

Protected Member Functions

- virtual Real computeQpResidual ()

  Required residual function for standard kernels in MOOSE.
- virtual Real computeQpJacobian ()

  Required Jacobian function for standard kernels in MOOSE.
- virtual Real computeQpOffDiagJacobian (unsigned int jvar)

  Not Required, but aids in the preconditioning step.

Protected Attributes

- bool _gaining

  Value is true if the time coef is positive.
- Real _time_coef

  Time coefficient for the coupled time derivative.
- const VariableValue & _coupled_dot

  Time derivative of the coupled variable.
- const VariableValue & _coupled_ddot

  Cross derivative term for the coupled variables.
- const unsigned int _coupled_var

  Variable identification for the coupled variable.

4.3.1 Detailed Description

CoupledCoeffTimeDerivative class object inherits from Kernel object.

This class object inherits from the Kernel object in the MOOSE framework. All public and protected members of this class are required function overrides. The kernel interfaces the two non-linear variables to couple a time derivative function between given objects.

Definition at line 53 of file CoupledCoeffTimeDerivative.h.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 CoupledCoeffTimeDerivative::CoupledCoeffTimeDerivative (const InputParameters & *parameters*)

Required constructor for objects in MOOSE.

4.3.3 Member Function Documentation

4.3.3.1 virtual Real CoupledCoeffTimeDerivative::computeQpJacobian ( ) [protected], [virtual]

Required Jacobian function for standard kernels in MOOSE.

This function returns a Jacobian contribution for this object. The Jacobian being computed is the associated diagonal element in the overall Jacobian matrix for the system and is used in preconditioning of the linear sub-problem.

4.3.3.2 virtual Real CoupledCoeffTimeDerivative::computeQpOffDiagJacobian (unsigned int *jvar*) [protected], [virtual]

Not Required, but aids in the preconditioning step.

This function returns the off diagonal Jacobian contribution for this object. By returning a non-zero value we will hopefully improve the convergence rate for the cross coupling of the variables.

4.3.3.3 virtual Real CoupledCoeffTimeDerivative::computeQpResidual ( ) [protected], [virtual]

Required residual function for standard kernels in MOOSE.

This function returns a residual contribution for this object.

4.3.4 Member Data Documentation

4.3.4.1 const VariableValue& CoupledCoeffTimeDerivative::_coupled_ddot [protected]

Cross derivative term for the coupled variables.

Definition at line 78 of file CoupledCoeffTimeDerivative.h.

4.3.4.2 const VariableValue& CoupledCoeffTimeDerivative::_coupled_dot [protected]

Time derivative of the coupled variable.

Definition at line 77 of file CoupledCoeffTimeDerivative.h.

4.3.4.3 const unsigned int CoupledCoeffTimeDerivative::_coupled_var [protected]

Variable identification for the coupled variable.

Definition at line 79 of file CoupledCoeffTimeDerivative.h.

4.3.4.4 bool CoupledCoeffTimeDerivative::_gaining [protected]

Value is true if the time coef is positive.

Definition at line 75 of file CoupledCoeffTimeDerivative.h.

4.3.4.5 Real CoupledCoeffTimeDerivative::_time_coef [protected]

Time coefficient for the coupled time derivative.

Definition at line 76 of file CoupledCoeffTimeDerivative.h.

The documentation for this class was generated from the following file:

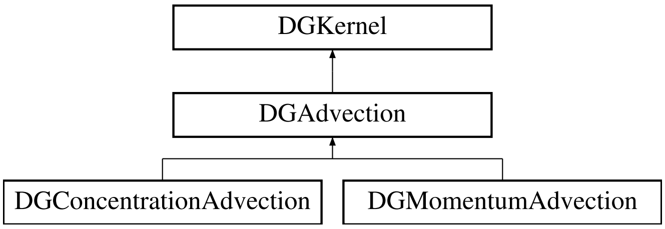
- CoupledCoeffTimeDerivative.h

4.4 DGAdvection Class Reference

DGAdvection class object inherits from DGKernel object.

```
#include <DGAdvection.h>
```

Inheritance diagram for DGAdvection:



Public Member Functions

- DGAdvection (const InputParameters &parameters)

  Required constructor for objects in MOOSE.

Protected Member Functions

- virtual Real computeQpResidual (Moose::DGResidualType type)

  Required residual function for DG kernels in MOOSE.
- virtual Real computeQpJacobian (Moose::DGJacobianType type)

  Required Jacobian function for DG kernels in MOOSE.

Protected Attributes

- RealVectorValue _velocity

  Vector of velocity.
- Real _vx

  x-component of velocity (optional - set in input file)
- Real _vy

  y-component of velocity (optional - set in input file)
- Real _vz

  z-component of velocity (optional - set in input file)

4.4.1 Detailed Description

DGAdvection class object inherits from DGKernel object.

This class object inherits from the DGKernel object in the MOOSE framework. All public and protected members of this class are required function overrides. The object will provide residuals and Jacobians for the discontinous Galerkin formulation of advection physics in the MOOSE framework. The only parameter for this kernel is a generic velocity vector, whose components can be set piecewise in the input file or by inheriting from this base class and manually altering the velocity vector.

Note

> As a reminder, any DGKernel in MOOSE was be accompanied by the equivalent GKernel in order to provide the full residuals and Jacobians for the system.

Definition at line 66 of file DGAdvection.h.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 DGAdvection::DGAdvection (const InputParameters & *parameters*)

Required constructor for objects in MOOSE.

4.4.3 Member Function Documentation

4.4.3.1 virtual Real DGAdvection::computeQpJacobian (Moose::DGJacobianType *type*) [protected], [virtual]

Required Jacobian function for DG kernels in MOOSE.

This function returns a Jacobian contribution for this object. The Jacobian being computed is the associated diagonal element in the overall Jacobian matrix for the system and is used in preconditioning of the linear sub-problem.

Reimplemented in DGConcentrationAdvection, and DGMomentumAdvection.

4.4.3.2 virtual Real DGAdvection::computeQpResidual (Moose::DGResidualType *type*) [protected], [virtual]

Required residual function for DG kernels in MOOSE.

This function returns a residual contribution for this object.

Reimplemented in DGConcentrationAdvection, and DGMomentumAdvection.

4.4.4 Member Data Documentation

4.4.4.1 RealVectorValue DGAdvection::_velocity [protected]

Vector of velocity.

Definition at line 82 of file DGAdvection.h.

4.4.4.2 Real DGAdvection::_vx [protected]

x-component of velocity (optional - set in input file)

Definition at line 83 of file DGAdvection.h.

4.4.4.3 Real DGAdvection::_vy [protected]

y-component of velocity (optional - set in input file)

Definition at line 84 of file DGAdvection.h.

4.4.4.4 Real DGAdvection::_vz [protected]

z-component of velocity (optional - set in input file)

Definition at line 85 of file DGAdvection.h.

The documentation for this class was generated from the following file:

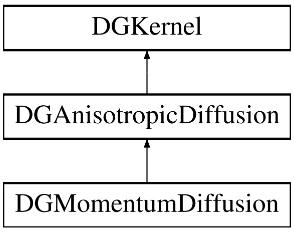
- DGAdvection.h

4.5 DGAnisotropicDiffusion Class Reference

DGAnisotropicDiffusion class object inherits from DGKernel object.

`#include <DGAnisotropicDiffusion.h>`

Inheritance diagram for DGAnisotropicDiffusion:



Public Member Functions

- DGAnisotropicDiffusion (const InputParameters &parameters)

    Required constructor for objects in MOOSE.

Protected Member Functions

- virtual Real computeQpResidual (Moose::DGResidualType type)

    Required residual function for DG kernels in MOOSE.
- virtual Real computeQpJacobian (Moose::DGJacobianType type)

    Required Jacobian function for DG kernels in MOOSE.

Protected Attributes

- Real _epsilon

    Penalty term for gradient jumps between the solution and test functions.
- Real _sigma

    Penalty term applied to element size.
- RealTensorValue _Diffusion

    Diffusion tensor matrix parameter.
- Real _Dxx
- Real _Dxy
- Real _Dxz
- Real _Dyx
- Real _Dyy
- Real _Dyz
- Real _Dzx
- Real _Dzy
- Real _Dzz

4.5.1 Detailed Description

DGAnisotropicDiffusion class object inherits from DGKernel object.

This class object inherits from the DGKernel object in the MOOSE framework. All public and protected members of this class are required function overrides. The object will provide residuals and Jacobians for the discontinous Galerkin formulation of advection physics in the MOOSE framework. The only parameter for this kernel is a diffusion tensor, whose components can be set piecewise in the input file or by inheriting from this base class and manually altering the tensor matrix.

Note

> As a reminder, any DGKernel in MOOSE was be accompanied by the equivalent GKernel in order to provide the full residuals and Jacobians for the system.

Definition at line 67 of file DGAnisotropicDiffusion.h.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 DGAnisotropicDiffusion::DGAnisotropicDiffusion (const InputParameters & *parameters*)

Required constructor for objects in MOOSE.

4.5.3 Member Function Documentation

4.5.3.1 virtual Real DGAnisotropicDiffusion::computeQpJacobian (Moose::DGJacobianType *type*) [protected], [virtual]

Required Jacobian function for DG kernels in MOOSE.

This function returns a Jacobian contribution for this object. The Jacobian being computed is the associated diagonal element in the overall Jacobian matrix for the system and is used in preconditioning of the linear sub-problem.

Reimplemented in DGMomentumDiffusion.

4.5.3.2 virtual Real DGAnisotropicDiffusion::computeQpResidual (Moose::DGResidualType *type*) [protected], [virtual]

Required residual function for DG kernels in MOOSE.

This function returns a residual contribution for this object.

Reimplemented in DGMomentumDiffusion.

4.5.4 Member Data Documentation

4.5.4.1 RealTensorValue DGAnisotropicDiffusion::_Diffusion [protected]

Diffusion tensor matrix parameter.

Definition at line 85 of file DGAnisotropicDiffusion.h.

4.5.4.2 Real DGAnisotropicDiffusion::_Dxx [protected]

Definition at line 87 of file DGAnisotropicDiffusion.h.

4.5.4.3 Real DGAnisotropicDiffusion::_Dxy [protected]

Definition at line 87 of file DGAnisotropicDiffusion.h.

4.5.4.4 Real DGAnisotropicDiffusion::_Dxz [protected]

Definition at line 87 of file DGAnisotropicDiffusion.h.

4.5.4.5 Real DGAnisotropicDiffusion::_Dyx [protected]

Definition at line 88 of file DGAnisotropicDiffusion.h.

4.5.4.6 Real DGAnisotropicDiffusion::_Dyy [protected]

Definition at line 88 of file DGAnisotropicDiffusion.h.

4.5.4.7 Real DGAnisotropicDiffusion::_Dyz [protected]

Definition at line 88 of file DGAnisotropicDiffusion.h.

4.5.4.8 Real DGAnisotropicDiffusion::_Dzx [protected]

Definition at line 89 of file DGAnisotropicDiffusion.h.

4.5.4.9 Real DGAnisotropicDiffusion::_Dzy [protected]

Definition at line 89 of file DGAnisotropicDiffusion.h.

4.5.4.10 Real DGAnisotropicDiffusion::_Dzz [protected]

Definition at line 89 of file DGAnisotropicDiffusion.h.

4.5.4.11 Real DGAnisotropicDiffusion::_epsilon [protected]

Penalty term for gradient jumps between the solution and test functions.

Definition at line 83 of file DGAnisotropicDiffusion.h.

4.5.4.12 Real DGAnisotropicDiffusion::_sigma [protected]

Penalty term applied to element size.

Definition at line 84 of file DGAnisotropicDiffusion.h.

The documentation for this class was generated from the following file:

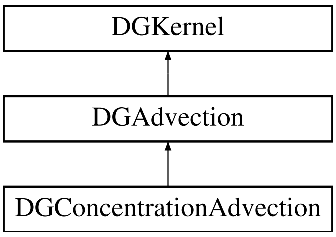
- DGAnisotropicDiffusion.h

4.6 DGConcentrationAdvection Class Reference

DGConcentrationAdvection class object inherits from DGKernel object.

```
#include <DGConcentrationAdvection.h>
```

Inheritance diagram for DGConcentrationAdvection:



Public Member Functions

- DGConcentrationAdvection (const InputParameters &parameters)

  Required constructor for objects in MOOSE.

Protected Member Functions

- virtual Real computeQpResidual (Moose::DGResidualType type)

  Required residual function for DG kernels in MOOSE.
- virtual Real computeQpJacobian (Moose::DGJacobianType type)

  Required Jacobian function for DG kernels in MOOSE.
- virtual Real computeQpOffDiagJacobian (Moose::DGJacobianType type, unsigned int jvar)

  Not required, but recomended function for DG kernels in MOOSE.

Protected Attributes

- const VariableValue & _ux

  Velocity in the x-direction.
- const VariableValue & _uy

  Velocity in the y-direction.
- const VariableValue & _uz

  Velocity in the z-direction.
- const unsigned int _ux_var

  Variable identification for ux.
- const unsigned int _uy_var

  Variable identification for uy.
- const unsigned int _uz_var

  Variable identification for uz.
- RealVectorValue _velocity

  Vector of velocity.
- Real _vx

  x-component of velocity (optional - set in input file)
- Real _vy

  y-component of velocity (optional - set in input file)
- Real _vz

  z-component of velocity (optional - set in input file)

4.6.1 Detailed Description

DGConcentrationAdvection class object inherits from DGKernel object.

This class object inherits from the DGKernel object in the MOOSE framework. All public and protected members of this class are required function overrides. The object will provide residuals and Jacobians for the discontinous Galerkin formulation of advection physics in the MOOSE framework. The only parameter for this kernel is a generic velocity vector, whose components can be set piecewise in the input file or by inheriting from this base class and manually altering the velocity vector.

Note

> As a reminder, any DGKernel in MOOSE was be accompanied by the equivalent GKernel in order to provide the full residuals and Jacobians for the system.

Definition at line 63 of file DGConcentrationAdvection.h.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 DGConcentrationAdvection::DGConcentrationAdvection (const InputParameters & *parameters*)

Required constructor for objects in MOOSE.

4.6.3 Member Function Documentation

4.6.3.1 virtual Real DGConcentrationAdvection::computeQpJacobian (Moose::DGJacobianType *type*) `[protected]`, `[virtual]`

Required Jacobian function for DG kernels in MOOSE.

This function returns a Jacobian contribution for this object. The Jacobian being computed is the associated diagonal element in the overall Jacobian matrix for the system and is used in preconditioning of the linear sub-problem.

Reimplemented from DGAdvection.

4.6.3.2 virtual Real DGConcentrationAdvection::computeQpOffDiagJacobian (Moose::DGJacobianType *type,* unsigned int *jvar*) `[protected]`, `[virtual]`

Not required, but recomended function for DG kernels in MOOSE.

This function returns an off-diagonal jacobian contribution for this object. The jacobian being computed will be associated with the variables coupled to this object and not the main coupled variable itself.

4.6.3.3 virtual Real DGConcentrationAdvection::computeQpResidual (Moose::DGResidualType *type*) `[protected]`, `[virtual]`

Required residual function for DG kernels in MOOSE.

This function returns a residual contribution for this object.

Reimplemented from DGAdvection.

4.6.4 Member Data Documentation

4.6.4.1 const VariableValue& DGConcentrationAdvection::_ux [protected]

Velocity in the x-direction.

Definition at line 86 of file DGConcentrationAdvection.h.

4.6.4.2 const unsigned int DGConcentrationAdvection::_ux_var [protected]

Variable identification for ux.

Definition at line 90 of file DGConcentrationAdvection.h.

4.6.4.3 const VariableValue& DGConcentrationAdvection::_uy [protected]

Velocity in the y-direction.

Definition at line 87 of file DGConcentrationAdvection.h.

4.6.4.4 const unsigned int DGConcentrationAdvection::_uy_var [protected]

Variable identification for uy.

Definition at line 91 of file DGConcentrationAdvection.h.

4.6.4.5 const VariableValue& DGConcentrationAdvection::_uz [protected]

Velocity in the z-direction.

Definition at line 88 of file DGConcentrationAdvection.h.

4.6.4.6 const unsigned int DGConcentrationAdvection::_uz_var [protected]

Variable identification for uz.

Definition at line 92 of file DGConcentrationAdvection.h.

4.6.4.7 RealVectorValue DGAdvection::_velocity [protected], [inherited]

Vector of velocity.

Definition at line 82 of file DGAdvection.h.

4.6.4.8 Real DGAdvection::_vx [protected], [inherited]

x-component of velocity (optional - set in input file)

Definition at line 83 of file DGAdvection.h.

4.6.4.9 Real DGAdvection::_vy [protected], [inherited]

y-component of velocity (optional - set in input file)

Definition at line 84 of file DGAdvection.h.

4.6.4.10 Real DGAdvection::_vz [protected], [inherited]

z-component of velocity (optional - set in input file)

Definition at line 85 of file DGAdvection.h.

The documentation for this class was generated from the following file:

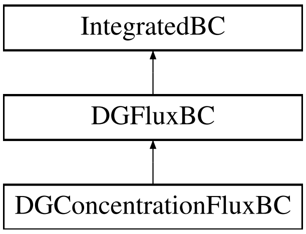
- DGConcentrationAdvection.h

4.7 DGConcentrationFluxBC Class Reference

DGConcentrationFluxBC class object inherits from IntegratedBC object.

```
#include <DGConcentrationFluxBC.h>
```

Inheritance diagram for DGConcentrationFluxBC:



Public Member Functions

- DGConcentrationFluxBC (const InputParameters &parameters)

  Required constructor for BC objects in MOOSE.

Protected Member Functions

- virtual Real computeQpResidual ()

  Required function override for BC objects in MOOSE.
- virtual Real computeQpJacobian ()

  Required function override for BC objects in MOOSE.
- virtual Real computeQpOffDiagJacobian (unsigned int jvar)

  Not Required, but aids in the preconditioning step.

Protected Attributes

- const VariableValue & _ux

  Velocity in the x-direction.
- const VariableValue & _uy

  Velocity in the y-direction.
- const VariableValue & _uz

  Velocity in the z-direction.
- const unsigned int _ux_var

  Variable identification for ux.
- const unsigned int _uy_var

  Variable identification for uy.
- const unsigned int _uz_var

  Variable identification for uz.
- RealVectorValue _velocity

  Velocity vector in the system or at the boundary.
- Real _vx
- Real _vy
- Real _vz
- Real _u_input

  Value of the non-linear variable at the input of the boundary.

4.7.1 Detailed Description

DGConcentrationFluxBC class object inherits from IntegratedBC object.

This class object inherits from the IntegratedBC object. All public and protected members of this class are required function overrides. The flux BC uses the velocity in the system to apply a boundary condition based on whether or not material is leaving or entering the boundary.

Definition at line 56 of file DGConcentrationFluxBC.h.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 DGConcentrationFluxBC::DGConcentrationFluxBC (const InputParameters & *parameters*)

Required constructor for BC objects in MOOSE.

4.7.3 Member Function Documentation

4.7.3.1 virtual Real DGConcentrationFluxBC::computeQpJacobian ( ) [protected], [virtual]

Required function override for BC objects in MOOSE.

This function returns a Jacobian contribution for this object. The Jacobian being computed is the associated diagonal element in the overall Jacobian matrix for the system and is used in preconditioning of the linear sub-problem.

Reimplemented from DGFluxBC.

4.7.3.2 virtual Real DGConcentrationFluxBC::computeQpOffDiagJacobian (unsigned int *jvar*) [protected], [virtual]

Not Required, but aids in the preconditioning step.

This function returns the off diagonal Jacobian contribution for this object. By returning a non-zero value we will hopefully improve the convergence rate for the cross coupling of the variables.

4.7.3.3 virtual Real DGConcentrationFluxBC::computeQpResidual ( ) [protected], [virtual]

Required function override for BC objects in MOOSE.

This function returns a residual contribution for this object.

Reimplemented from DGFluxBC.

4.7.4 Member Data Documentation

4.7.4.1 Real DGFluxBC::_u_input [protected], [inherited]

Value of the non-linear variable at the input of the boundary.

Definition at line 87 of file DGFluxBC.h.

4.7.4.2 const VariableValue& DGConcentrationFluxBC::_ux [protected]

Velocity in the x-direction.

Definition at line 79 of file DGConcentrationFluxBC.h.

4.7.4.3 const unsigned int DGConcentrationFluxBC::_ux_var [protected]

Variable identification for ux.

Definition at line 83 of file DGConcentrationFluxBC.h.

4.7.4.4 const VariableValue& DGConcentrationFluxBC::_uy [protected]

Velocity in the y-direction.

Definition at line 80 of file DGConcentrationFluxBC.h.

4.7.4.5 const unsigned int DGConcentrationFluxBC::_uy_var [protected]

Variable identification for uy.

Definition at line 84 of file DGConcentrationFluxBC.h.

4.7.4.6 const VariableValue& DGConcentrationFluxBC::_uz [protected]

Velocity in the z-direction.

Definition at line 81 of file DGConcentrationFluxBC.h.

4.7.4.7 const unsigned int DGConcentrationFluxBC::_uz_var [protected]

Variable identification for uz.

Definition at line 85 of file DGConcentrationFluxBC.h.

4.7.4.8 RealVectorValue DGFluxBC::_velocity [protected], [inherited]

Velocity vector in the system or at the boundary.

Definition at line 80 of file DGFluxBC.h.

4.7.4.9 Real DGFluxBC::_vx [protected], [inherited]

Definition at line 82 of file DGFluxBC.h.

4.7.4.10 Real DGFluxBC::_vy [protected], [inherited]

Definition at line 83 of file DGFluxBC.h.

4.7.4.11 Real DGFluxBC::_vz [protected], [inherited]

Definition at line 84 of file DGFluxBC.h.

The documentation for this class was generated from the following file:

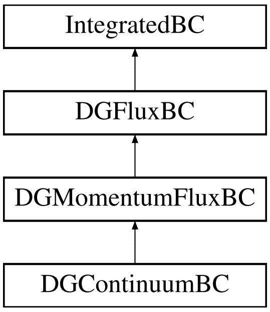
- DGConcentrationFluxBC.h

4.8 DGContinuumBC Class Reference

DGMomentumFluxBC class object inherits from IntegratedBC object.

```
#include <DGContinuumBC.h>
```

Inheritance diagram for DGContinuumBC:



Public Member Functions

- DGContinuumBC (const InputParameters &parameters)

  Required constructor for BC objects in MOOSE.

Protected Member Functions

- virtual Real computeQpResidual ()

  Required function override for BC objects in MOOSE.
- virtual Real computeQpJacobian ()

  Required function override for BC objects in MOOSE.
- virtual Real computeQpOffDiagJacobian (unsigned int jvar)

  Not Required, but aids in the preconditioning step.

Protected Attributes

- const VariableValue & _ux

  Velocity in the x-direction.
- const VariableValue & _uy

  Velocity in the y-direction.
- const VariableValue & _uz

  Velocity in the z-direction.
- const VariableValue & _density

  Density of the fluid.
- unsigned int _dir

  Direction variable for direction this kernel acts on (0=x, 1=y, 2=z)
- const unsigned int _ux_var

  Variable identification for ux.
- const unsigned int _uy_var

  Variable identification for uy.
- const unsigned int _uz_var

  Variable identification for uz.
- const unsigned int _den_var

  Variable identification for density.
- RealVectorValue _velocity

  Velocity vector in the system or at the boundary.
- Real _vx
- Real _vy
- Real _vz
- Real _u_input

  Value of the non-linear variable at the input of the boundary.

4.8.1 Detailed Description

DGMomentumFluxBC class object inherits from IntegratedBC object.

This class object inherits from the IntegratedBC object. All public and protected members of this class are required function overrides. The flux BC uses the velocity in the system to apply a boundary condition based on whether or not material is leaving or entering the boundary.

Definition at line 53 of file DGContinuumBC.h.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 DGContinuumBC::DGContinuumBC (const InputParameters & *parameters*)

Required constructor for BC objects in MOOSE.

4.8.3 Member Function Documentation

4.8.3.1 virtual Real DGContinuumBC::computeQpJacobian ( ) [protected], [virtual]

Required function override for BC objects in MOOSE.

This function returns a Jacobian contribution for this object. The Jacobian being computed is the associated diagonal element in the overall Jacobian matrix for the system and is used in preconditioning of the linear sub-problem.

Reimplemented from DGMomentumFluxBC.

4.8.3.2 virtual Real DGContinuumBC::computeQpOffDiagJacobian (unsigned int *jvar*) [protected], [virtual]

Not Required, but aids in the preconditioning step.

This function returns the off diagonal Jacobian contribution for this object. By returning a non-zero value we will hopefully improve the convergence rate for the cross coupling of the variables.

Reimplemented from DGMomentumFluxBC.

4.8.3.3 virtual Real DGContinuumBC::computeQpResidual ( ) [protected], [virtual]

Required function override for BC objects in MOOSE.

This function returns a residual contribution for this object.

Reimplemented from DGMomentumFluxBC.

4.8.4 Member Data Documentation

4.8.4.1 const unsigned int DGMomentumFluxBC::_den_var [protected], [inherited]

Variable identification for density.

Definition at line 90 of file DGMomentumFluxBC.h.

4.8.4.2 const VariableValue& DGMomentumFluxBC::_density [protected], [inherited]

Density of the fluid.

Definition at line 83 of file DGMomentumFluxBC.h.

4.8.4.3 unsigned int DGMomentumFluxBC::_dir [protected], [inherited]

Direction variable for direction this kernel acts on (0=x, 1=y, 2=z)

Definition at line 85 of file DGMomentumFluxBC.h.

4.8.4.4 Real DGFluxBC::_u_input [protected], [inherited]

Value of the non-linear variable at the input of the boundary.

Definition at line 87 of file DGFluxBC.h.

4.8.4.5 const VariableValue& DGMomentumFluxBC::_ux [protected], [inherited]

Velocity in the x-direction.

Definition at line 79 of file DGMomentumFluxBC.h.

4.8.4.6 const unsigned int DGMomentumFluxBC::_ux_var [protected], [inherited]

Variable identification for ux.

Definition at line 87 of file DGMomentumFluxBC.h.

4.8.4.7 const VariableValue& DGMomentumFluxBC::_uy [protected], [inherited]

Velocity in the y-direction.

Definition at line 80 of file DGMomentumFluxBC.h.

4.8.4.8 const unsigned int DGMomentumFluxBC::_uy_var [protected], [inherited]

Variable identification for uy.

Definition at line 88 of file DGMomentumFluxBC.h.

4.8.4.9 const VariableValue& DGMomentumFluxBC::_uz [protected], [inherited]

Velocity in the z-direction.

Definition at line 81 of file DGMomentumFluxBC.h.

4.8.4.10 const unsigned int DGMomentumFluxBC::_uz_var [protected], [inherited]

Variable identification for uz.

Definition at line 89 of file DGMomentumFluxBC.h.

4.8.4.11 RealVectorValue DGFluxBC::_velocity [protected], [inherited]

Velocity vector in the system or at the boundary.

Definition at line 80 of file DGFluxBC.h.

4.8.4.12 Real DGFluxBC::_vx [protected], [inherited]

Definition at line 82 of file DGFluxBC.h.

4.8.4.13 Real DGFluxBC::_vy [protected], [inherited]

Definition at line 83 of file DGFluxBC.h.

4.8.4.14 Real DGFluxBC::_vz [protected], [inherited]

Definition at line 84 of file DGFluxBC.h.

The documentation for this class was generated from the following file:

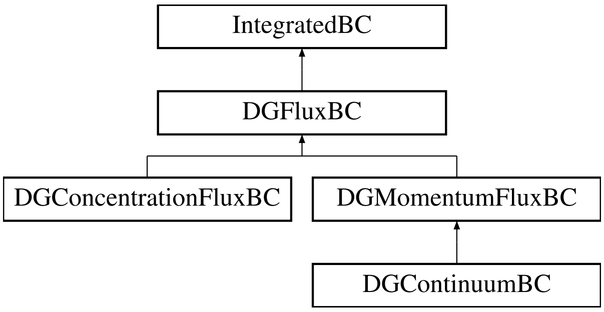
- DGContinuumBC.h

4.9 DGFluxBC Class Reference

DGFluxBC class object inherits from IntegratedBC object.

```
#include <DGFluxBC.h>
```

Inheritance diagram for DGFluxBC:



Public Member Functions

- DGFluxBC (const InputParameters &parameters)

  Required constructor for BC objects in MOOSE.

Protected Member Functions

- virtual Real computeQpResidual ()

  Required function override for BC objects in MOOSE.
- virtual Real computeQpJacobian ()

  Required function override for BC objects in MOOSE.

Protected Attributes

- RealVectorValue _velocity

  Velocity vector in the system or at the boundary.
- Real _vx
- Real _vy
- Real _vz
- Real _u_input

  Value of the non-linear variable at the input of the boundary.

4.9.1 Detailed Description

DGFluxBC class object inherits from IntegratedBC object.

This class object inherits from the IntegratedBC object. All public and protected members of this class are required function overrides. The flux BC uses the velocity in the system to apply a boundary condition based on whether or not material is leaving or entering the boundary.

Definition at line 63 of file DGFluxBC.h.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 DGFluxBC::DGFluxBC (const InputParameters & *parameters*)

Required constructor for BC objects in MOOSE.

4.9.3 Member Function Documentation

4.9.3.1 virtual Real DGFluxBC::computeQpJacobian ( ) `[protected]`, `[virtual]`

Required function override for BC objects in MOOSE.

This function returns a Jacobian contribution for this object. The Jacobian being computed is the associated diagonal element in the overall Jacobian matrix for the system and is used in preconditioning of the linear sub-problem.

Reimplemented in DGConcentrationFluxBC, DGMomentumFluxBC, and DGContinuumBC.

4.9.3.2 virtual Real DGFluxBC::computeQpResidual ( ) `[protected]`, `[virtual]`

Required function override for BC objects in MOOSE.

This function returns a residual contribution for this object.

Reimplemented in DGConcentrationFluxBC, DGMomentumFluxBC, and DGContinuumBC.

4.9.4 Member Data Documentation

4.9.4.1 Real DGFluxBC::_u_input `[protected]`

Value of the non-linear variable at the input of the boundary.

Definition at line 87 of file DGFluxBC.h.

4.9.4.2 RealVectorValue DGFluxBC::_velocity [protected]

Velocity vector in the system or at the boundary.

Definition at line 80 of file DGFluxBC.h.

4.9.4.3 Real DGFluxBC::_vx [protected]

Definition at line 82 of file DGFluxBC.h.

4.9.4.4 Real DGFluxBC::_vy [protected]

Definition at line 83 of file DGFluxBC.h.

4.9.4.5 Real DGFluxBC::_vz [protected]

Definition at line 84 of file DGFluxBC.h.

The documentation for this class was generated from the following file:

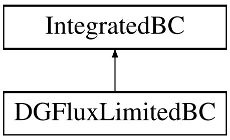
- DGFluxBC.h

4.10 DGFluxLimitedBC Class Reference

DGFluxLimitedBC class object inherits from IntegratedBC object.

```
#include <DGFluxLimitedBC.h>
```

Inheritance diagram for DGFluxLimitedBC:



Public Member Functions

- DGFluxLimitedBC (const InputParameters &parameters)

  Required constructor for BC objects in MOOSE.

Protected Member Functions

- virtual Real computeQpResidual ()

  Required function override for BC objects in MOOSE.
- virtual Real computeQpJacobian ()

  Required function override for BC objects in MOOSE.

Protected Attributes

- Real _epsilon

  Penalty term applied to the difference between the solution at the inlet and the value it is supposed to be.
- Real _sigma

  Penalty term based on the size of the element at the boundary.
- RealVectorValue _velocity

  Velocity vector in the system or at the boundary.
- RealTensorValue _Diffusion

  Diffusivity tensory in the system or at the boundary.
- Real _vx
- Real _vy
- Real _vz
- Real _Dxx
- Real _Dxy
- Real _Dxz
- Real _Dyx
- Real _Dyy
- Real _Dyz
- Real _Dzx
- Real _Dzy
- Real _Dzz
- Real _u_input

  Value of the non-linear variable at the input of the boundary.

4.10.1 Detailed Description

DGFluxLimitedBC class object inherits from IntegratedBC object.

This class object inherits from the IntegratedBC object. All public and protected members of this class are required function overrides.

Definition at line 56 of file DGFluxLimitedBC.h.

4.10.2 Constructor & Destructor Documentation

4.10.2.1 DGFluxLimitedBC::DGFluxLimitedBC (const InputParameters & *parameters*)

Required constructor for BC objects in MOOSE.

4.10.3 Member Function Documentation

4.10.3.1 virtual Real DGFluxLimitedBC::computeQpJacobian ( ) [protected], [virtual]

Required function override for BC objects in MOOSE.

This function returns a Jacobian contribution for this object. The Jacobian being computed is the associated diagonal element in the overall Jacobian matrix for the system and is used in preconditioning of the linear sub-problem.

4.10.3.2 virtual Real DGFluxLimitedBC::computeQpResidual ( ) [protected], [virtual]

Required function override for BC objects in MOOSE.

This function returns a residual contribution for this object.

4.10.4 Member Data Documentation

4.10.4.1 RealTensorValue DGFluxLimitedBC::_Diffusion [protected]

Diffusivity tensory in the system or at the boundary.

Definition at line 80 of file DGFluxLimitedBC.h.

4.10.4.2 Real DGFluxLimitedBC::_Dxx [protected]

Definition at line 86 of file DGFluxLimitedBC.h.

4.10.4.3 Real DGFluxLimitedBC::_Dxy [protected]

Definition at line 86 of file DGFluxLimitedBC.h.

4.10.4.4 Real DGFluxLimitedBC::_Dxz [protected]

Definition at line 86 of file DGFluxLimitedBC.h.

4.10.4.5 Real DGFluxLimitedBC::_Dyx [protected]

Definition at line 87 of file DGFluxLimitedBC.h.

4.10.4.6 Real DGFluxLimitedBC::_Dyy [protected]

Definition at line 87 of file DGFluxLimitedBC.h.

4.10.4.7 Real DGFluxLimitedBC::_Dyz [protected]

Definition at line 87 of file DGFluxLimitedBC.h.

4.10.4.8 Real DGFluxLimitedBC::_Dzx [protected]

Definition at line 88 of file DGFluxLimitedBC.h.

4.10.4.9 Real DGFluxLimitedBC::_Dzy [protected]

Definition at line 88 of file DGFluxLimitedBC.h.

4.10.4.10 Real DGFluxLimitedBC::_Dzz [protected]

Definition at line 88 of file DGFluxLimitedBC.h.

4.10.4.11 Real DGFluxLimitedBC::_epsilon [protected]

Penalty term applied to the difference between the solution at the inlet and the value it is supposed to be.

Definition at line 73 of file DGFluxLimitedBC.h.

4.10.4.12 Real DGFluxLimitedBC::_sigma [protected]

Penalty term based on the size of the element at the boundary.

Definition at line 75 of file DGFluxLimitedBC.h.

4.10.4.13 Real DGFluxLimitedBC::_u_input [protected]

Value of the non-linear variable at the input of the boundary.

Definition at line 91 of file DGFluxLimitedBC.h.

4.10.4.14 RealVectorValue DGFluxLimitedBC::_velocity [protected]

Velocity vector in the system or at the boundary.

Definition at line 78 of file DGFluxLimitedBC.h.

4.10.4.15 Real DGFluxLimitedBC::_vx [protected]

Definition at line 82 of file DGFluxLimitedBC.h.

4.10.4.16 Real DGFluxLimitedBC::_vy [protected]

Definition at line 83 of file DGFluxLimitedBC.h.

4.10.4.17 Real DGFluxLimitedBC::_vz [protected]

Definition at line 84 of file DGFluxLimitedBC.h.

The documentation for this class was generated from the following file:

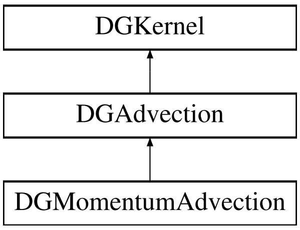
- DGFluxLimitedBC.h

4.11 DGMomentumAdvection Class Reference

DGMomentumAdvection class object inherits from DGKernel object.

```
#include <DGMomentumAdvection.h>
```

Inheritance diagram for DGMomentumAdvection:



Public Member Functions

- DGMomentumAdvection (const InputParameters &parameters)

  Required constructor for objects in MOOSE.

Protected Member Functions

- virtual Real computeQpResidual (Moose::DGResidualType type)

  Required residual function for DG kernels in MOOSE.
- virtual Real computeQpJacobian (Moose::DGJacobianType type)

  Required Jacobian function for DG kernels in MOOSE.
- virtual Real computeQpOffDiagJacobian (Moose::DGJacobianType type, unsigned int jvar)

  Not required, but recomended function for DG kernels in MOOSE.

Protected Attributes

- const VariableValue & _ux

  Velocity in the x-direction.
- const VariableValue & _uy

  Velocity in the y-direction.
- const VariableValue & _uz

  Velocity in the z-direction.
- const VariableValue & _density

  Density of the fluid.
- unsigned int _dir

  Direction variable for direction this kernel acts on (0=x, 1=y, 2=z)
- const unsigned int _ux_var

  Variable identification for ux.
- const unsigned int _uy_var

  Variable identification for uy.
- const unsigned int _uz_var

  Variable identification for uz.
- const unsigned int _den_var

  Variable identification for density.
- RealVectorValue _velocity

  Vector of velocity.
- Real _vx

  x-component of velocity (optional - set in input file)
- Real _vy

  y-component of velocity (optional - set in input file)
- Real _vz

  z-component of velocity (optional - set in input file)

4.11.1 Detailed Description

DGMomentumAdvection class object inherits from DGKernel object.

This class object inherits from the DGKernel object in the MOOSE framework. All public and protected members of this class are required function overrides. The object will provide residuals and Jacobians for the discontinous Galerkin formulation of advection physics in the MOOSE framework. The only parameter for this kernel is a generic velocity vector, whose components can be set piecewise in the input file or by inheriting from this base class and manually altering the velocity vector.

Note

> As a reminder, any DGKernel in MOOSE was be accompanied by the equivalent GKernel in order to provide the full residuals and Jacobians for the system.

Definition at line 63 of file DGMomentumAdvection.h.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 DGMomentumAdvection::DGMomentumAdvection (const InputParameters & *parameters*)

Required constructor for objects in MOOSE.

4.11.3 Member Function Documentation

4.11.3.1 virtual Real DGMomentumAdvection::computeQpJacobian (Moose::DGJacobianType *type*) `[protected]`, `[virtual]`

Required Jacobian function for DG kernels in MOOSE.

This function returns a Jacobian contribution for this object. The Jacobian being computed is the associated diagonal element in the overall Jacobian matrix for the system and is used in preconditioning of the linear sub-problem.

Reimplemented from DGAdvection.

4.11.3.2 virtual Real DGMomentumAdvection::computeQpOffDiagJacobian (Moose::DGJacobianType *type,* unsigned int *jvar*) `[protected]`, `[virtual]`

Not required, but recomended function for DG kernels in MOOSE.

This function returns an off-diagonal jacobian contribution for this object. The jacobian being computed will be associated with the variables coupled to this object and not the main coupled variable itself.

4.11.3.3 virtual Real DGMomentumAdvection::computeQpResidual (Moose::DGResidualType *type*) `[protected]`, `[virtual]`

Required residual function for DG kernels in MOOSE.

This function returns a residual contribution for this object.

Reimplemented from DGAdvection.

4.11.4 Member Data Documentation

4.11.4.1 const unsigned int DGMomentumAdvection::_den_var [protected]

Variable identification for density.

Definition at line 97 of file DGMomentumAdvection.h.

4.11.4.2 const VariableValue& DGMomentumAdvection::_density [protected]

Density of the fluid.

Definition at line 90 of file DGMomentumAdvection.h.

4.11.4.3 unsigned int DGMomentumAdvection::_dir [protected]

Direction variable for direction this kernel acts on (0=x, 1=y, 2=z)

Definition at line 92 of file DGMomentumAdvection.h.

4.11.4.4 const VariableValue& DGMomentumAdvection::_ux [protected]

Velocity in the x-direction.

Definition at line 86 of file DGMomentumAdvection.h.

4.11.4.5 const unsigned int DGMomentumAdvection::_ux_var [protected]

Variable identification for ux.

Definition at line 94 of file DGMomentumAdvection.h.

4.11.4.6 const VariableValue& DGMomentumAdvection::_uy [protected]

Velocity in the y-direction.

Definition at line 87 of file DGMomentumAdvection.h.

4.11.4.7 const unsigned int DGMomentumAdvection::_uy_var [protected]

Variable identification for uy.

Definition at line 95 of file DGMomentumAdvection.h.

4.11.4.8 const VariableValue& DGMomentumAdvection::_uz [protected]

Velocity in the z-direction.

Definition at line 88 of file DGMomentumAdvection.h.

4.11.4.9 const unsigned int DGMomentumAdvection::_uz_var [protected]

Variable identification for uz.

Definition at line 96 of file DGMomentumAdvection.h.

4.11.4.10 RealVectorValue DGAdvection::_velocity [protected], [inherited]

Vector of velocity.

Definition at line 82 of file DGAdvection.h.

4.11.4.11 Real DGAdvection::_vx [protected], [inherited]

x-component of velocity (optional - set in input file)

Definition at line 83 of file DGAdvection.h.

4.11.4.12 Real DGAdvection::_vy [protected], [inherited]

y-component of velocity (optional - set in input file)

Definition at line 84 of file DGAdvection.h.

4.11.4.13 Real DGAdvection::_vz [protected], [inherited]

z-component of velocity (optional - set in input file)

Definition at line 85 of file DGAdvection.h.

The documentation for this class was generated from the following file:

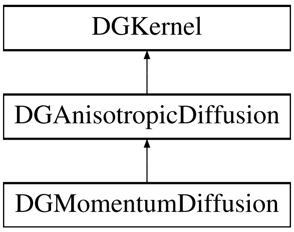
- DGMomentumAdvection.h

4.12 DGMomentumDiffusion Class Reference

DGMomentumDiffusion class object inherits from DGKernel object.

```
#include <DGMomentumDiffusion.h>
```

Inheritance diagram for DGMomentumDiffusion:



Public Member Functions

- DGMomentumDiffusion (const InputParameters &parameters)

    Required constructor for objects in MOOSE.

Protected Member Functions

- virtual Real computeQpResidual (Moose::DGResidualType type)

    Required residual function for DG kernels in MOOSE.
- virtual Real computeQpJacobian (Moose::DGJacobianType type)

    Required Jacobian function for DG kernels in MOOSE.
- virtual Real computeQpOffDiagJacobian (Moose::DGJacobianType type, unsigned int jvar)

    Not required, but recomended function for DG kernels in MOOSE.

Protected Attributes

- const VariableValue & _viscosity

    Viscosity of the fluid.
- const unsigned int _vis_var

    Variable identification for viscosity.
- Real _epsilon

    Penalty term for gradient jumps between the solution and test functions.
- Real _sigma

    Penalty term applied to element size.
- RealTensorValue _Diffusion

    Diffusion tensor matrix parameter.
- Real _Dxx
- Real _Dxy
- Real _Dxz
- Real _Dyx
- Real _Dyy
- Real _Dyz
- Real _Dzx
- Real _Dzy
- Real _Dzz

4.12.1 Detailed Description

DGMomentumDiffusion class object inherits from DGKernel object.

This class object inherits from the DGKernel object in the MOOSE framework. All public and protected members of this class are required function overrides. The object will provide residuals and Jacobians for the discontinous Galerkin formulation of advection physics in the MOOSE framework.

Note

> As a reminder, any DGKernel in MOOSE was be accompanied by the equivalent GKernel in order to provide the full residuals and Jacobians for the system.

Definition at line 61 of file DGMomentumDiffusion.h.

4.12.2 Constructor & Destructor Documentation

4.12.2.1 DGMomentumDiffusion::DGMomentumDiffusion (const InputParameters & *parameters*)

Required constructor for objects in MOOSE.

4.12.3 Member Function Documentation

4.12.3.1 virtual Real DGMomentumDiffusion::computeQpJacobian (Moose::DGJacobianType *type*) [protected], [virtual]

Required Jacobian function for DG kernels in MOOSE.

This function returns a Jacobian contribution for this object. The Jacobian being computed is the associated diagonal element in the overall Jacobian matrix for the system and is used in preconditioning of the linear sub-problem.

Reimplemented from DGAnisotropicDiffusion.

4.12.3.2 virtual Real DGMomentumDiffusion::computeQpOffDiagJacobian (Moose::DGJacobianType *type,* unsigned int *jvar*) [protected], [virtual]

Not required, but recomended function for DG kernels in MOOSE.

This function returns an off-diagonal jacobian contribution for this object. The jacobian being computed will be associated with the variables coupled to this object and not the main coupled variable itself.

4.12.3.3 virtual Real DGMomentumDiffusion::computeQpResidual (Moose::DGResidualType *type*) [protected], [virtual]

Required residual function for DG kernels in MOOSE.

This function returns a residual contribution for this object.

Reimplemented from DGAnisotropicDiffusion.

4.12.4 Member Data Documentation

4.12.4.1 RealTensorValue DGAnisotropicDiffusion::_Diffusion [protected], [inherited]

Diffusion tensor matrix parameter.

Definition at line 85 of file DGAnisotropicDiffusion.h.

4.12.4.2 Real DGAnisotropicDiffusion::_Dxx [protected], [inherited]

Definition at line 87 of file DGAnisotropicDiffusion.h.

4.12.4.3 Real DGAnisotropicDiffusion::_Dxy [protected], [inherited]

Definition at line 87 of file DGAnisotropicDiffusion.h.

4.12.4.4 Real DGAnisotropicDiffusion::_Dxz [protected], [inherited]

Definition at line 87 of file DGAnisotropicDiffusion.h.

4.12.4.5 Real DGAnisotropicDiffusion::_Dyx [protected], [inherited]

Definition at line 88 of file DGAnisotropicDiffusion.h.

4.12.4.6 Real DGAnisotropicDiffusion::_Dyy [protected], [inherited]

Definition at line 88 of file DGAnisotropicDiffusion.h.

4.12.4.7 Real DGAnisotropicDiffusion::_Dyz [protected], [inherited]

Definition at line 88 of file DGAnisotropicDiffusion.h.

4.12.4.8 Real DGAnisotropicDiffusion::_Dzx [protected], [inherited]

Definition at line 89 of file DGAnisotropicDiffusion.h.

4.12.4.9 Real DGAnisotropicDiffusion::_Dzy [protected], [inherited]

Definition at line 89 of file DGAnisotropicDiffusion.h.

4.12.4.10 Real DGAnisotropicDiffusion::_Dzz [protected], [inherited]

Definition at line 89 of file DGAnisotropicDiffusion.h.

4.12.4.11 Real DGAnisotropicDiffusion::_epsilon [protected], [inherited]

Penalty term for gradient jumps between the solution and test functions.

Definition at line 83 of file DGAnisotropicDiffusion.h.

4.12.4.12 Real DGAnisotropicDiffusion::_sigma [protected], [inherited]

Penalty term applied to element size.

Definition at line 84 of file DGAnisotropicDiffusion.h.

4.12.4.13 const unsigned int DGMomentumDiffusion::_vis_var [protected]

Variable identification for viscosity.

Definition at line 85 of file DGMomentumDiffusion.h.

4.12.4.14 const VariableValue& DGMomentumDiffusion::_viscosity [protected]

Viscosity of the fluid.

Definition at line 84 of file DGMomentumDiffusion.h.

The documentation for this class was generated from the following file:

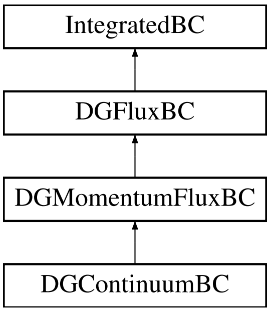
- DGMomentumDiffusion.h

4.13 DGMomentumFluxBC Class Reference

DGMomentumFluxBC class object inherits from IntegratedBC object.

`#include <DGMomentumFluxBC.h>`

Inheritance diagram for DGMomentumFluxBC:



Public Member Functions

- DGMomentumFluxBC (const InputParameters &parameters)

    Required constructor for BC objects in MOOSE.

Protected Member Functions

- virtual Real computeQpResidual ()

    Required function override for BC objects in MOOSE.
- virtual Real computeQpJacobian ()

    Required function override for BC objects in MOOSE.
- virtual Real computeQpOffDiagJacobian (unsigned int jvar)

    Not Required, but aids in the preconditioning step.

Protected Attributes

- const VariableValue & _ux

  Velocity in the x-direction.
- const VariableValue & _uy

  Velocity in the y-direction.
- const VariableValue & _uz

  Velocity in the z-direction.
- const VariableValue & _density

  Density of the fluid.
- unsigned int _dir

  Direction variable for direction this kernel acts on (0=x, 1=y, 2=z)
- const unsigned int _ux_var

  Variable identification for ux.
- const unsigned int _uy_var

  Variable identification for uy.
- const unsigned int _uz_var

  Variable identification for uz.
- const unsigned int _den_var

  Variable identification for density.
- RealVectorValue _velocity

  Velocity vector in the system or at the boundary.
- Real _vx
- Real _vy
- Real _vz
- Real _u_input

  Value of the non-linear variable at the input of the boundary.

4.13.1 Detailed Description

DGMomentumFluxBC class object inherits from IntegratedBC object.

This class object inherits from the IntegratedBC object. All public and protected members of this class are required function overrides. The flux BC uses the velocity in the system to apply a boundary condition based on whether or not material is leaving or entering the boundary.

Definition at line 56 of file DGMomentumFluxBC.h.

4.13.2 Constructor & Destructor Documentation

4.13.2.1 DGMomentumFluxBC::DGMomentumFluxBC (const InputParameters & *parameters*)

Required constructor for BC objects in MOOSE.

4.13.3 Member Function Documentation

4.13.3.1 virtual Real DGMomentumFluxBC::computeQpJacobian ( ) [protected], [virtual]

Required function override for BC objects in MOOSE.

This function returns a Jacobian contribution for this object. The Jacobian being computed is the associated diagonal element in the overall Jacobian matrix for the system and is used in preconditioning of the linear sub-problem.

Reimplemented from DGFluxBC.

Reimplemented in DGContinuumBC.

4.13.3.2 virtual Real DGMomentumFluxBC::computeQpOffDiagJacobian (unsigned int *jvar*) [protected], [virtual]

Not Required, but aids in the preconditioning step.

This function returns the off diagonal Jacobian contribution for this object. By returning a non-zero value we will hopefully improve the convergence rate for the cross coupling of the variables.

Reimplemented in DGContinuumBC.

4.13.3.3 virtual Real DGMomentumFluxBC::computeQpResidual ( ) [protected], [virtual]

Required function override for BC objects in MOOSE.

This function returns a residual contribution for this object.

Reimplemented from DGFluxBC.

Reimplemented in DGContinuumBC.

4.13.4 Member Data Documentation

4.13.4.1 const unsigned int DGMomentumFluxBC::_den_var [protected]

Variable identification for density.

Definition at line 90 of file DGMomentumFluxBC.h.

4.13.4.2 const VariableValue& DGMomentumFluxBC::_density [protected]

Density of the fluid.

Definition at line 83 of file DGMomentumFluxBC.h.

4.13.4.3 unsigned int DGMomentumFluxBC::_dir [protected]

Direction variable for direction this kernel acts on (0=x, 1=y, 2=z)

Definition at line 85 of file DGMomentumFluxBC.h.

4.13.4.4 Real DGFluxBC::_u_input [protected], [inherited]

Value of the non-linear variable at the input of the boundary.

Definition at line 87 of file DGFluxBC.h.

4.13.4.5 const VariableValue& DGMomentumFluxBC::_ux [protected]

Velocity in the x-direction.

Definition at line 79 of file DGMomentumFluxBC.h.

4.13.4.6 const unsigned int DGMomentumFluxBC::_ux_var [protected]

Variable identification for ux.

Definition at line 87 of file DGMomentumFluxBC.h.

4.13.4.7 const VariableValue& DGMomentumFluxBC::_uy [protected]

Velocity in the y-direction.

Definition at line 80 of file DGMomentumFluxBC.h.

4.13.4.8 const unsigned int DGMomentumFluxBC::_uy_var [protected]

Variable identification for uy.

Definition at line 88 of file DGMomentumFluxBC.h.

4.13.4.9 const VariableValue& DGMomentumFluxBC::_uz [protected]

Velocity in the z-direction.

Definition at line 81 of file DGMomentumFluxBC.h.

4.13.4.10 const unsigned int DGMomentumFluxBC::_uz_var [protected]

Variable identification for uz.

Definition at line 89 of file DGMomentumFluxBC.h.

4.13.4.11 RealVectorValue DGFluxBC::_velocity [protected], [inherited]

Velocity vector in the system or at the boundary.

Definition at line 80 of file DGFluxBC.h.

4.13.4.12 Real DGFluxBC::_vx [protected], [inherited]

Definition at line 82 of file DGFluxBC.h.

4.13.4.13 Real DGFluxBC::_vy [protected], [inherited]

Definition at line 83 of file DGFluxBC.h.

4.13.4.14 Real DGFluxBC::_vz [protected], [inherited]

Definition at line 84 of file DGFluxBC.h.

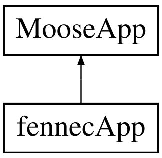
The documentation for this class was generated from the following file:

- DGMomentumFluxBC.h

4.14 fennecApp Class Reference

```
#include <fennecApp.h>
```

Inheritance diagram for fennecApp:



Public Member Functions

- fennecApp (InputParameters parameters)
- virtual ∼fennecApp ()

Static Public Member Functions

- static void registerApps ()
- static void registerObjects (Factory &factory)
- static void registerObjectDepends (Factory &factory)
- static void associateSyntax (Syntax &syntax, ActionFactory &action_factory)
- static void associateSyntaxDepends (Syntax &syntax, ActionFactory &action_factory)
- static void registerExecFlags (Factory &factory)

4.14.1 Detailed Description

Definition at line 19 of file fennecApp.h.

4.14.2 Constructor & Destructor Documentation

4.14.2.1 fennecApp::fennecApp (InputParameters *parameters*)

4.14.2.2 virtual fennecApp::∼fennecApp ( ) [virtual]

4.14.3 Member Function Documentation

4.14.3.1 static void fennecApp::associateSyntax (Syntax & *syntax,* ActionFactory & *action_factory*) [static]

4.14.3.2 static void fennecApp::associateSyntaxDepends (Syntax & *syntax,* ActionFactory & *action_factory*) [static]

4.14.3.3 static void fennecApp::registerApps ( ) [static]

4.14.3.4 static void fennecApp::registerExecFlags (Factory & *factory*) [static]

4.14.3.5 static void fennecApp::registerObjectDepends (Factory & *factory*) [static]

4.14.3.6 static void fennecApp::registerObjects (Factory & *factory*) [static]

The documentation for this class was generated from the following file:

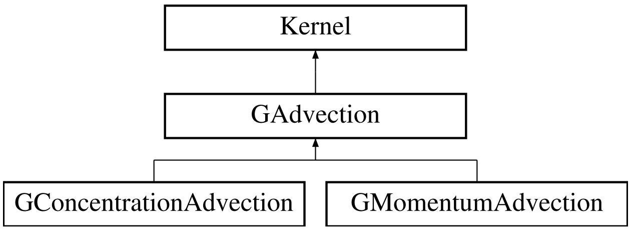
- fennecApp.h

4.15 GAdvection Class Reference

GAdvection class object inherits from Kernel object.

```
#include <GAdvection.h>
```

Inheritance diagram for GAdvection:



Public Member Functions

- GAdvection (const InputParameters &parameters)

  Required constructor for objects in MOOSE.

Protected Member Functions

- virtual Real computeQpResidual ()

  Required residual function for standard kernels in MOOSE.
- virtual Real computeQpJacobian ()

  Required Jacobian function for standard kernels in MOOSE.

Protected Attributes

- RealVectorValue _velocity

  Vector of velocity.
- Real _vx

  x-component of velocity (optional - set in input file)
- Real _vy

  y-component of velocity (optional - set in input file)
- Real _vz

  z-component of velocity (optional - set in input file)

4.15.1 Detailed Description

GAdvection class object inherits from Kernel object.

This class object inherits from the Kernel object in the MOOSE framework. All public and protected members of this class are required function overrides. The kernel has a velocity vector whose components can be set piecewise in an input file.

Note

> To create a specific GAdvection kernel, inherit from this class and override the components of the velocity vector, then call the residual and Jacobian functions for this object.

Definition at line 58 of file GAdvection.h.

4.15.2 Constructor & Destructor Documentation

4.15.2.1 GAdvection::GAdvection (const InputParameters & *parameters*)

Required constructor for objects in MOOSE.

4.15.3 Member Function Documentation

4.15.3.1 virtual Real GAdvection::computeQpJacobian ( ) [protected], [virtual]

Required Jacobian function for standard kernels in MOOSE.

This function returns a Jacobian contribution for this object. The Jacobian being computed is the associated diagonal element in the overall Jacobian matrix for the system and is used in preconditioning of the linear sub-problem.

Reimplemented in GConcentrationAdvection, and GMomentumAdvection.

4.15.3.2 virtual Real GAdvection::computeQpResidual ( ) [protected], [virtual]

Required residual function for standard kernels in MOOSE.

This function returns a residual contribution for this object.

Reimplemented in GConcentrationAdvection, and GMomentumAdvection.

4.15.4 Member Data Documentation

4.15.4.1 RealVectorValue GAdvection::_velocity [protected]

Vector of velocity.

Definition at line 74 of file GAdvection.h.

4.15.4.2 Real GAdvection::_vx [protected]

x-component of velocity (optional - set in input file)

Definition at line 76 of file GAdvection.h.

4.15.4.3 Real GAdvection::_vy [protected]

y-component of velocity (optional - set in input file)

Definition at line 77 of file GAdvection.h.

4.15.4.4 Real GAdvection::_vz [protected]

z-component of velocity (optional - set in input file)

Definition at line 78 of file GAdvection.h.

The documentation for this class was generated from the following file:

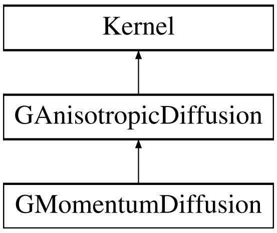
- GAdvection.h

4.16 GAnisotropicDiffusion Class Reference

GAnisotropicDiffusion class object inherits from Kernel object.

```
#include <GAnisotropicDiffusion.h>
```

Inheritance diagram for GAnisotropicDiffusion:



Public Member Functions

- GAnisotropicDiffusion (const InputParameters &parameters)

  Required constructor for objects in MOOSE.

Protected Member Functions

- virtual Real computeQpResidual ()

  Required residual function for standard kernels in MOOSE.
- virtual Real computeQpJacobian ()

  Required Jacobian function for standard kernels in MOOSE.

Protected Attributes

- RealTensorValue _Diffusion

  Diffusion tensor matrix parameter.
- Real _Dxx
- Real _Dxy
- Real _Dxz
- Real _Dyx
- Real _Dyy
- Real _Dyz
- Real _Dzx
- Real _Dzy
- Real _Dzz

4.16.1 Detailed Description

GAnisotropicDiffusion class object inherits from Kernel object.

This class object inherits from the Kernel object in the MOOSE framework. All public and protected members of this class are required function overrides. The kernel has a diffusion tensor whose components can be set piecewise in an input file.

Note

> To create a specific GAnisotropicDiffusion kernel, inherit from this class and override the components of the diffusion tensor, then call the residual and Jacobian functions for this object.

Definition at line 58 of file GAnisotropicDiffusion.h.

4.16.2 Constructor & Destructor Documentation

4.16.2.1 GAnisotropicDiffusion::GAnisotropicDiffusion (const InputParameters & *parameters*)

Required constructor for objects in MOOSE.

4.16.3 Member Function Documentation

4.16.3.1 virtual Real GAnisotropicDiffusion::computeQpJacobian ( ) [protected], [virtual]

Required Jacobian function for standard kernels in MOOSE.

This function returns a Jacobian contribution for this object. The Jacobian being computed is the associated diagonal element in the overall Jacobian matrix for the system and is used in preconditioning of the linear sub-problem.

Reimplemented in GMomentumDiffusion.

4.16.3.2 virtual Real GAnisotropicDiffusion::computeQpResidual ( ) [protected], [virtual]

Required residual function for standard kernels in MOOSE.

This function returns a residual contribution for this object.

Reimplemented in GMomentumDiffusion.

4.16.4 Member Data Documentation

4.16.4.1 RealTensorValue GAnisotropicDiffusion::_Diffusion [protected]

Diffusion tensor matrix parameter.

Definition at line 74 of file GAnisotropicDiffusion.h.

4.16.4.2 Real GAnisotropicDiffusion::_Dxx [protected]

Definition at line 76 of file GAnisotropicDiffusion.h.

4.16.4.3 Real GAnisotropicDiffusion::_Dxy [protected]

Definition at line 76 of file GAnisotropicDiffusion.h.

4.16.4.4 Real GAnisotropicDiffusion::_Dxz [protected]

Definition at line 76 of file GAnisotropicDiffusion.h.

4.16.4.5 Real GAnisotropicDiffusion::_Dyx [protected]

Definition at line 77 of file GAnisotropicDiffusion.h.

4.16.4.6 Real GAnisotropicDiffusion::_Dyy [protected]

Definition at line 77 of file GAnisotropicDiffusion.h.

4.16.4.7 Real GAnisotropicDiffusion::_Dyz [protected]

Definition at line 77 of file GAnisotropicDiffusion.h.

4.16.4.8 Real GAnisotropicDiffusion::_Dzx [protected]

Definition at line 78 of file GAnisotropicDiffusion.h.

4.16.4.9 Real GAnisotropicDiffusion::_Dzy [protected]

Definition at line 78 of file GAnisotropicDiffusion.h.

4.16.4.10 Real GAnisotropicDiffusion::_Dzz [protected]

Definition at line 78 of file GAnisotropicDiffusion.h.

The documentation for this class was generated from the following file:

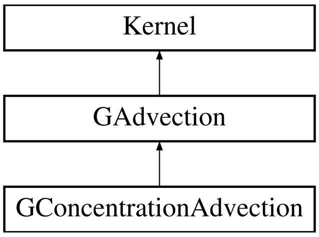
- GAnisotropicDiffusion.h

4.17 GConcentrationAdvection Class Reference

GConcentrationAdvection class object inherits from Kernel object.

```
#include <GConcentrationAdvection.h>
```

Inheritance diagram for GConcentrationAdvection:



Public Member Functions

- GConcentrationAdvection (const InputParameters &parameters)

  Required constructor for objects in MOOSE.

Protected Member Functions

- virtual Real computeQpResidual ()

  Required residual function for standard kernels in MOOSE.
- virtual Real computeQpJacobian ()

  Required Jacobian function for standard kernels in MOOSE.
- virtual Real computeQpOffDiagJacobian (unsigned int jvar)

  Not Required, but aids in the preconditioning step.

Protected Attributes

- const VariableValue & _ux

  Velocity in the x-direction.
- const VariableValue & _uy

  Velocity in the y-direction.
- const VariableValue & _uz

  Velocity in the z-direction.
- const unsigned int _ux_var

  Variable identification for ux.
- const unsigned int _uy_var

  Variable identification for uy.
- const unsigned int _uz_var

  Variable identification for uz.
- RealVectorValue _velocity

  Vector of velocity.
- Real _vx

  x-component of velocity (optional - set in input file)
- Real _vy

  y-component of velocity (optional - set in input file)
- Real _vz

  z-component of velocity (optional - set in input file)

4.17.1 Detailed Description

GConcentrationAdvection class object inherits from Kernel object.

This class object inherits from the Kernel object in the MOOSE framework. All public and protected members of this class are required function overrides. The kernel has a velocity vector whose components can be set piecewise in an input file.

Note

> To create a specific GAdvection kernel, inherit from this class and override the components of the velocity vector, then call the residual and Jacobian functions for this object.

Definition at line 58 of file GConcentrationAdvection.h.

4.17.2 Constructor & Destructor Documentation

4.17.2.1 GConcentrationAdvection::GConcentrationAdvection (const InputParameters & *parameters*)

Required constructor for objects in MOOSE.

4.17.3 Member Function Documentation

4.17.3.1 virtual Real GConcentrationAdvection::computeQpJacobian ( ) [protected], [virtual]

Required Jacobian function for standard kernels in MOOSE.

This function returns a Jacobian contribution for this object. The Jacobian being computed is the associated diagonal element in the overall Jacobian matrix for the system and is used in preconditioning of the linear sub-problem.

Reimplemented from GAdvection.

4.17.3.2 virtual Real GConcentrationAdvection::computeQpOffDiagJacobian (unsigned int *jvar*) [protected], [virtual]

Not Required, but aids in the preconditioning step.

This function returns the off diagonal Jacobian contribution for this object. By returning a non-zero value we will hopefully improve the convergence rate for the cross coupling of the variables.

4.17.3.3 virtual Real GConcentrationAdvection::computeQpResidual ( ) [protected], [virtual]

Required residual function for standard kernels in MOOSE.

This function returns a residual contribution for this object.

Reimplemented from GAdvection.

4.17.4 Member Data Documentation

4.17.4.1 const VariableValue& GConcentrationAdvection::_ux [protected]

Velocity in the x-direction.

Definition at line 81 of file GConcentrationAdvection.h.

4.17.4.2 const unsigned int GConcentrationAdvection::_ux_var [protected]

Variable identification for ux.

Definition at line 85 of file GConcentrationAdvection.h.

4.17.4.3 const VariableValue& GConcentrationAdvection::_uy [protected]

Velocity in the y-direction.

Definition at line 82 of file GConcentrationAdvection.h.

4.17.4.4 const unsigned int GConcentrationAdvection::_uy_var [protected]

Variable identification for uy.

Definition at line 86 of file GConcentrationAdvection.h.

4.17.4.5 const VariableValue& GConcentrationAdvection::_uz [protected]

Velocity in the z-direction.

Definition at line 83 of file GConcentrationAdvection.h.

4.17.4.6 const unsigned int GConcentrationAdvection::_uz_var [protected]

Variable identification for uz.

Definition at line 87 of file GConcentrationAdvection.h.

4.17.4.7 RealVectorValue GAdvection::_velocity [protected], [inherited]

Vector of velocity.

Definition at line 74 of file GAdvection.h.

4.17.4.8 Real GAdvection::_vx [protected], [inherited]

x-component of velocity (optional - set in input file)

Definition at line 76 of file GAdvection.h.

4.17.4.9 Real GAdvection::_vy [protected], [inherited]

y-component of velocity (optional - set in input file)

Definition at line 77 of file GAdvection.h.

4.17.4.10 Real GAdvection::_vz [protected], [inherited]

z-component of velocity (optional - set in input file)

Definition at line 78 of file GAdvection.h.

The documentation for this class was generated from the following file:

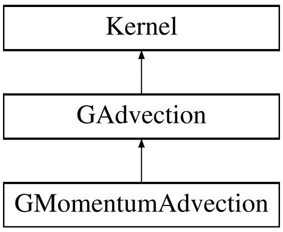
- GConcentrationAdvection.h

4.18 GMomentumAdvection Class Reference

GAdvection class object inherits from Kernel object.

```
#include <GMomentumAdvection.h>
```

Inheritance diagram for GMomentumAdvection:



Public Member Functions

- GMomentumAdvection (const InputParameters &parameters)

    Required constructor for objects in MOOSE.

Protected Member Functions

- virtual Real computeQpResidual ()

  Required residual function for standard kernels in MOOSE.
- virtual Real computeQpJacobian ()

  Required Jacobian function for standard kernels in MOOSE.
- virtual Real computeQpOffDiagJacobian (unsigned int jvar)

  Not Required, but aids in the preconditioning step.

Protected Attributes

- const VariableValue & _ux

  Velocity in the x-direction.
- const VariableValue & _uy

  Velocity in the y-direction.
- const VariableValue & _uz

  Velocity in the z-direction.
- const VariableValue & _density

  Density of the fluid.
- unsigned int _dir

  Direction variable for direction this kernel acts on (0=x, 1=y, 2=z)
- const unsigned int _ux_var

  Variable identification for ux.
- const unsigned int _uy_var

  Variable identification for uy.
- const unsigned int _uz_var

  Variable identification for uz.
- const unsigned int _den_var

  Variable identification for density.
- RealVectorValue _velocity

  Vector of velocity.
- Real _vx

  x-component of velocity (optional - set in input file)
- Real _vy

  y-component of velocity (optional - set in input file)
- Real _vz

  z-component of velocity (optional - set in input file)

4.18.1 Detailed Description

GAdvection class object inherits from Kernel object.

This class object inherits from the Kernel object in the MOOSE framework. All public and protected members of this class are required function overrides. The kernel has a velocity vector whose components can be set piecewise in an input file.

Note

> To create a specific GAdvection kernel, inherit from this class and override the components of the velocity vector, then call the residual and Jacobian functions for this object.

Definition at line 58 of file GMomentumAdvection.h.

4.18.2 Constructor & Destructor Documentation

4.18.2.1 GMomentumAdvection::GMomentumAdvection (const InputParameters & *parameters*)

Required constructor for objects in MOOSE.

4.18.3 Member Function Documentation

4.18.3.1 virtual Real GMomentumAdvection::computeQpJacobian ( ) [protected], [virtual]

Required Jacobian function for standard kernels in MOOSE.

This function returns a Jacobian contribution for this object. The Jacobian being computed is the associated diagonal element in the overall Jacobian matrix for the system and is used in preconditioning of the linear sub-problem.

Reimplemented from GAdvection.

4.18.3.2 virtual Real GMomentumAdvection::computeQpOffDiagJacobian (unsigned int *jvar*) [protected], [virtual]

Not Required, but aids in the preconditioning step.

This function returns the off diagonal Jacobian contribution for this object. By returning a non-zero value we will hopefully improve the convergence rate for the cross coupling of the variables.

4.18.3.3 virtual Real GMomentumAdvection::computeQpResidual ( ) [protected], [virtual]

Required residual function for standard kernels in MOOSE.

This function returns a residual contribution for this object.

Reimplemented from GAdvection.

4.18.4 Member Data Documentation

4.18.4.1 const unsigned int GMomentumAdvection::_den_var [protected]

Variable identification for density.

Definition at line 92 of file GMomentumAdvection.h.

4.18.4.2 const VariableValue& GMomentumAdvection::_density [protected]

Density of the fluid.

Definition at line 85 of file GMomentumAdvection.h.

4.18.4.3 unsigned int GMomentumAdvection::_dir [protected]

Direction variable for direction this kernel acts on (0=x, 1=y, 2=z)

Definition at line 87 of file GMomentumAdvection.h.

4.18.4.4 const VariableValue& GMomentumAdvection::_ux [protected]

Velocity in the x-direction.

Definition at line 81 of file GMomentumAdvection.h.

4.18.4.5 const unsigned int GMomentumAdvection::_ux_var [protected]

Variable identification for ux.

Definition at line 89 of file GMomentumAdvection.h.

4.18.4.6 const VariableValue& GMomentumAdvection::_uy [protected]

Velocity in the y-direction.

Definition at line 82 of file GMomentumAdvection.h.

4.18.4.7 const unsigned int GMomentumAdvection::_uy_var [protected]

Variable identification for uy.

Definition at line 90 of file GMomentumAdvection.h.

4.18.4.8 const VariableValue& GMomentumAdvection::_uz [protected]

Velocity in the z-direction.

Definition at line 83 of file GMomentumAdvection.h.

4.18.4.9 const unsigned int GMomentumAdvection::_uz_var [protected]

Variable identification for uz.

Definition at line 91 of file GMomentumAdvection.h.

4.18.4.10 RealVectorValue GAdvection::_velocity [protected], [inherited]

Vector of velocity.

Definition at line 74 of file GAdvection.h.

4.18.4.11 Real GAdvection::_vx [protected], [inherited]

x-component of velocity (optional - set in input file)

Definition at line 76 of file GAdvection.h.

4.18.4.12 Real GAdvection::_vy [protected], [inherited]

y-component of velocity (optional - set in input file)

Definition at line 77 of file GAdvection.h.

4.18.4.13 Real GAdvection::_vz [protected], [inherited]

z-component of velocity (optional - set in input file)

Definition at line 78 of file GAdvection.h.

The documentation for this class was generated from the following file:

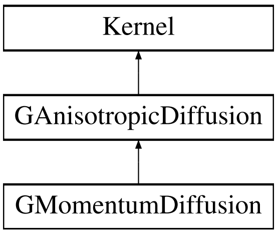
- GMomentumAdvection.h

4.19 GMomentumDiffusion Class Reference

GAnisotropicDiffusion class object inherits from Kernel object.

```
#include <GMomentumDiffusion.h>
```

Inheritance diagram for GMomentumDiffusion:



Public Member Functions

- GMomentumDiffusion (const InputParameters &parameters)

  Required constructor for objects in MOOSE.

Protected Member Functions

- virtual Real computeQpResidual ()

  Required residual function for standard kernels in MOOSE.
- virtual Real computeQpJacobian ()

  Required Jacobian function for standard kernels in MOOSE.
- virtual Real computeQpOffDiagJacobian (unsigned int jvar)

  Not Required, but aids in the preconditioning step.

Protected Attributes

- const VariableValue & _viscosity

  Viscosity of the fluid.
- const unsigned int _vis_var

  Variable identification for viscosity.
- RealTensorValue _Diffusion

  Diffusion tensor matrix parameter.
- Real _Dxx
- Real _Dxy
- Real _Dxz
- Real _Dyx
- Real _Dyy
- Real _Dyz
- Real _Dzx
- Real _Dzy
- Real _Dzz

4.19.1 Detailed Description

GAnisotropicDiffusion class object inherits from Kernel object.

This class object inherits from the Kernel object in the MOOSE framework. All public and protected members of this class are required function overrides. The kernel has a diffusion tensor whose components can be set piecewise in an input file.

Note

> To create a specific GMomentumDiffusion kernel, inherit from this class and override the components of the diffusion tensor, then call the residual and Jacobian functions for this object.

Definition at line 56 of file GMomentumDiffusion.h.

4.19.2 Constructor & Destructor Documentation

4.19.2.1 GMomentumDiffusion::GMomentumDiffusion (const InputParameters & *parameters*)

Required constructor for objects in MOOSE.

4.19.3 Member Function Documentation

4.19.3.1 virtual Real GMomentumDiffusion::computeQpJacobian ( ) `[protected], [virtual]`

Required Jacobian function for standard kernels in MOOSE.

This function returns a Jacobian contribution for this object. The Jacobian being computed is the associated diagonal element in the overall Jacobian matrix for the system and is used in preconditioning of the linear sub-problem.

Reimplemented from GAnisotropicDiffusion.

4.19.3.2 virtual Real GMomentumDiffusion::computeQpOffDiagJacobian (unsigned int *jvar*) `[protected], [virtual]`

Not Required, but aids in the preconditioning step.

This function returns the off diagonal Jacobian contribution for this object. By returning a non-zero value we will hopefully improve the convergence rate for the cross coupling of the variables.

4.19.3.3 virtual Real GMomentumDiffusion::computeQpResidual ( ) `[protected], [virtual]`

Required residual function for standard kernels in MOOSE.

This function returns a residual contribution for this object.

Reimplemented from GAnisotropicDiffusion.

4.19.4 Member Data Documentation

4.19.4.1 RealTensorValue GAnisotropicDiffusion::_Diffusion [protected], [inherited]

Diffusion tensor matrix parameter.

Definition at line 74 of file GAnisotropicDiffusion.h.

4.19.4.2 Real GAnisotropicDiffusion::_Dxx [protected], [inherited]

Definition at line 76 of file GAnisotropicDiffusion.h.

4.19.4.3 Real GAnisotropicDiffusion::_Dxy [protected], [inherited]

Definition at line 76 of file GAnisotropicDiffusion.h.

4.19.4.4 Real GAnisotropicDiffusion::_Dxz [protected], [inherited]

Definition at line 76 of file GAnisotropicDiffusion.h.

4.19.4.5 Real GAnisotropicDiffusion::_Dyx [protected], [inherited]

Definition at line 77 of file GAnisotropicDiffusion.h.

4.19.4.6 Real GAnisotropicDiffusion::_Dyy [protected], [inherited]

Definition at line 77 of file GAnisotropicDiffusion.h.

4.19.4.7 Real GAnisotropicDiffusion::_Dyz [protected], [inherited]

Definition at line 77 of file GAnisotropicDiffusion.h.

4.19.4.8 Real GAnisotropicDiffusion::_Dzx [protected], [inherited]

Definition at line 78 of file GAnisotropicDiffusion.h.

4.19.4.9 Real GAnisotropicDiffusion::_Dzy [protected], [inherited]

Definition at line 78 of file GAnisotropicDiffusion.h.

4.19.4.10 Real GAnisotropicDiffusion::_Dzz [protected], [inherited]

Definition at line 78 of file GAnisotropicDiffusion.h.

4.19.4.11 const unsigned int GMomentumDiffusion::_vis_var [protected]

Variable identification for viscosity.

Definition at line 80 of file GMomentumDiffusion.h.

4.19.4.12 const VariableValue& GMomentumDiffusion::_viscosity [protected]

Viscosity of the fluid.

Definition at line 79 of file GMomentumDiffusion.h.

The documentation for this class was generated from the following file:

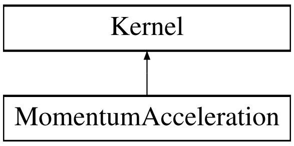
- GMomentumDiffusion.h

4.20 MomentumAcceleration Class Reference

MomentumAcceleration class object inherits from Kernel object.

```
#include <MomentumAcceleration.h>
```

Inheritance diagram for MomentumAcceleration:



Public Member Functions

- MomentumAcceleration (const InputParameters &parameters)

  Required constructor for objects in MOOSE.

Protected Member Functions

- virtual Real computeQpResidual ()

  Required residual function for standard kernels in MOOSE.
- virtual Real computeQpJacobian ()

  Required Jacobian function for standard kernels in MOOSE.
- virtual Real computeQpOffDiagJacobian (unsigned int jvar)

  Not Required, but aids in the preconditioning step.

Protected Attributes

- const VariableValue & _density

  Density of the fluid.
- const VariableValue & _accel

  Variable for acceleration in the vector direction the kernel acts on.
- const unsigned int _den_var

  Variable identification for density.
- const unsigned int _accel_var

  Variable identification for acceleration.

4.20.1 Detailed Description

MomentumAcceleration class object inherits from Kernel object.

This class object inherits from the Kernel object in the MOOSE framework. All public and protected members of this class are required function overrides. The kernel couples acceleration and density variables to a momentum balance for either x, y, or z components of fluid velocity.

Definition at line 55 of file MomentumAcceleration.h.

4.20.2 Constructor & Destructor Documentation

4.20.2.1 MomentumAcceleration::MomentumAcceleration (const InputParameters & *parameters*)

Required constructor for objects in MOOSE.

4.20.3 Member Function Documentation

4.20.3.1 virtual Real MomentumAcceleration::computeQpJacobian ( ) `[protected]`, `[virtual]`

Required Jacobian function for standard kernels in MOOSE.

This function returns a Jacobian contribution for this object. The Jacobian being computed is the associated diagonal element in the overall Jacobian matrix for the system and is used in preconditioning of the linear sub-problem.

4.20.3.2 virtual Real MomentumAcceleration::computeQpOffDiagJacobian (unsigned int *jvar*) `[protected]`, `[virtual]`

Not Required, but aids in the preconditioning step.

This function returns the off diagonal Jacobian contribution for this object. By returning a non-zero value we will hopefully improve the convergence rate for the cross coupling of the variables.

4.20.3.3 virtual Real MomentumAcceleration::computeQpResidual ( ) `[protected]`, `[virtual]`

Required residual function for standard kernels in MOOSE.

This function returns a residual contribution for this object.

4.20.4 Member Data Documentation

4.20.4.1 const VariableValue& MomentumAcceleration::_accel `[protected]`

Variable for acceleration in the vector direction the kernel acts on.

Definition at line 79 of file MomentumAcceleration.h.

4.20.4.2 const unsigned int MomentumAcceleration::_accel_var [protected]

Variable identification for acceleration.

Definition at line 82 of file MomentumAcceleration.h.

4.20.4.3 const unsigned int MomentumAcceleration::_den_var [protected]

Variable identification for density.

Definition at line 81 of file MomentumAcceleration.h.

4.20.4.4 const VariableValue& MomentumAcceleration::_density [protected]

Density of the fluid.

Definition at line 78 of file MomentumAcceleration.h.

The documentation for this class was generated from the following file:

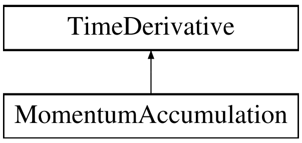
- MomentumAcceleration.h

4.21 MomentumAccumulation Class Reference

MomentumAccumulation class object inherits from TimeDerivative object.

```
#include <MomentumAccumulation.h>
```

Inheritance diagram for MomentumAccumulation:



Public Member Functions

- MomentumAccumulation (const InputParameters &parameters)

  Required constructor for objects in MOOSE.

Protected Member Functions

- virtual Real computeQpResidual ()

  Required residual function for standard kernels in MOOSE.
- virtual Real computeQpJacobian ()

  Required Jacobian function for standard kernels in MOOSE.
- virtual Real computeQpOffDiagJacobian (unsigned int jvar)

  Not Required, but aids in the preconditioning step.

Protected Attributes

- const VariableValue & _density

  Density of the fluid.
- const unsigned int _den_var

  Variable identification for density.

4.21.1 Detailed Description

MomentumAccumulation class object inherits from TimeDerivative object.

This class object inherits from the TimeDerivative object. All public and protected members of this class are required function overrides. The kernel couples with the density of the medium and calls the standard TimeDerivative functions while appending the density variable to those values.

Definition at line 52 of file MomentumAccumulation.h.

4.21.2 Constructor & Destructor Documentation

4.21.2.1 MomentumAccumulation::MomentumAccumulation (const InputParameters & *parameters*)

Required constructor for objects in MOOSE.

4.21.3 Member Function Documentation

4.21.3.1 virtual Real MomentumAccumulation::computeQpJacobian ( ) `[protected], [virtual]`

Required Jacobian function for standard kernels in MOOSE.

This function returns a Jacobian contribution for this object. The Jacobian being computed is the associated diagonal element in the overall Jacobian matrix for the system and is used in preconditioning of the linear sub-problem.

4.21.3.2 virtual Real MomentumAccumulation::computeQpOffDiagJacobian (unsigned int *jvar*) `[protected], [virtual]`

Not Required, but aids in the preconditioning step.

This function returns the off diagonal Jacobian contribution for this object. By returning a non-zero value we will hopefully improve the convergence rate for the cross coupling of the variables.

4.21.3.3 virtual Real MomentumAccumulation::computeQpResidual ( ) `[protected], [virtual]`

Required residual function for standard kernels in MOOSE.

This function returns a residual contribution for this object.

4.21.4 Member Data Documentation

4.21.4.1 const unsigned int MomentumAccumulation::_den_var [protected]

Variable identification for density.

Definition at line 77 of file MomentumAccumulation.h.

4.21.4.2 const VariableValue& MomentumAccumulation::_density [protected]

Density of the fluid.

Definition at line 75 of file MomentumAccumulation.h.

The documentation for this class was generated from the following file:

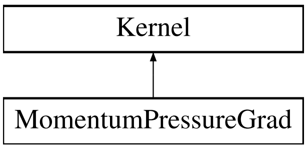
- MomentumAccumulation.h

4.22 MomentumPressureGrad Class Reference

MomentumPressureGrad class object inherits from Kernel object.

```
#include <MomentumPressureGrad.h>
```

Inheritance diagram for MomentumPressureGrad:



Public Member Functions

- MomentumPressureGrad (const InputParameters &parameters)

  Required constructor for objects in MOOSE.

Protected Member Functions

- virtual Real computeQpResidual ()

  Required residual function for standard kernels in MOOSE.
- virtual Real computeQpJacobian ()

  Required Jacobian function for standard kernels in MOOSE.
- virtual Real computeQpOffDiagJacobian (unsigned int jvar)

  Not Required, but aids in the preconditioning step.

Protected Attributes

- const VariableGradient & _press_grad

  Pressure gradient of the fluid.
- unsigned int _dir

  Direction variable for direction this kernel acts on (0=x, 1=y, 2=z)
- const unsigned int _press_var

  Variable identification for density.

4.22.1 Detailed Description

MomentumPressureGrad class object inherits from Kernel object.

This class object inherits from the Kernel object in the MOOSE framework. All public and protected members of this class are required function overrides. The kernel couples a pressure gradient variable to a momentum balance for either x, y, or z components of fluid velocity.

Definition at line 56 of file MomentumPressureGrad.h.

4.22.2 Constructor & Destructor Documentation

4.22.2.1 MomentumPressureGrad::MomentumPressureGrad (const InputParameters & *parameters*)

Required constructor for objects in MOOSE.

4.22.3 Member Function Documentation

4.22.3.1 virtual Real MomentumPressureGrad::computeQpJacobian ( ) `[protected]`, `[virtual]`

Required Jacobian function for standard kernels in MOOSE.

This function returns a Jacobian contribution for this object. The Jacobian being computed is the associated diagonal element in the overall Jacobian matrix for the system and is used in preconditioning of the linear sub-problem.

4.22.3.2 virtual Real MomentumPressureGrad::computeQpOffDiagJacobian (unsigned int *jvar*) `[protected]`, `[virtual]`

Not Required, but aids in the preconditioning step.

This function returns the off diagonal Jacobian contribution for this object. By returning a non-zero value we will hopefully improve the convergence rate for the cross coupling of the variables.

4.22.3.3 virtual Real MomentumPressureGrad::computeQpResidual ( ) `[protected]`, `[virtual]`

Required residual function for standard kernels in MOOSE.

This function returns a residual contribution for this object.

4.22.4 Member Data Documentation

4.22.4.1 unsigned int MomentumPressureGrad::_dir [protected]

Direction variable for direction this kernel acts on (0=x, 1=y, 2=z)

Definition at line 81 of file MomentumPressureGrad.h.

4.22.4.2 const VariableGradient& MomentumPressureGrad::_press_grad [protected]

Pressure gradient of the fluid.

Definition at line 79 of file MomentumPressureGrad.h.

4.22.4.3 const unsigned int MomentumPressureGrad::_press_var [protected]

Variable identification for density.

Definition at line 83 of file MomentumPressureGrad.h.

The documentation for this class was generated from the following file:

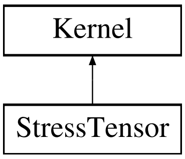
- MomentumPressureGrad.h

4.23 StressTensor Class Reference

StressTensor class object inherits from Kernel object.

`#include <StressTensor.h>`

Inheritance diagram for StressTensor:



Public Member Functions

- StressTensor (const InputParameters &parameters)

  Required constructor for objects in MOOSE.

Protected Member Functions

- virtual Real computeQpResidual ()

  Required residual function for standard kernels in MOOSE.
- virtual Real computeQpJacobian ()

  Required Jacobian function for standard kernels in MOOSE.
- virtual Real computeQpOffDiagJacobian (unsigned int jvar)

  Not Required, but aids in the preconditioning step.

Protected Attributes

- const VariableGradient & _ux_grad

  Velocity gradient for d(ux)/dx.
- const VariableGradient & _uy_grad

  Velocity gradient for d(uy)/dy.
- const VariableGradient & _uz_grad

  Velocity gradient for d(uz)/dz.
- const VariableValue & _viscosity

  Viscosity of the fluid.
- unsigned int _dir

  Direction variable for direction this kernel acts on (0=x, 1=y, 2=z)
- const unsigned int _ux_var

  Variable identification for ux.
- const unsigned int _uy_var

  Variable identification for uy.
- const unsigned int _uz_var

  Variable identification for uz.
- const unsigned int _vis_var

  Variable identification for viscosity.

4.23.1 Detailed Description

StressTensor class object inherits from Kernel object.

This class object inherits from the Kernel object in the MOOSE framework. All public and protected members of this class are required function overrides.

Definition at line 51 of file StressTensor.h.

4.23.2 Constructor & Destructor Documentation

4.23.2.1 StressTensor::StressTensor (const InputParameters & *parameters*)

Required constructor for objects in MOOSE.

4.23.3 Member Function Documentation

4.23.3.1 virtual Real StressTensor::computeQpJacobian ( ) [protected], [virtual]

Required Jacobian function for standard kernels in MOOSE.

This function returns a Jacobian contribution for this object. The Jacobian being computed is the associated diagonal element in the overall Jacobian matrix for the system and is used in preconditioning of the linear sub-problem.

4.23.3.2 virtual Real StressTensor::computeQpOffDiagJacobian (unsigned int *jvar*) [protected], [virtual]

Not Required, but aids in the preconditioning step.

This function returns the off diagonal Jacobian contribution for this object. By returning a non-zero value we will hopefully improve the convergence rate for the cross coupling of the variables.

4.23.3.3 virtual Real StressTensor::computeQpResidual ( ) [protected], [virtual]

Required residual function for standard kernels in MOOSE.

This function returns a residual contribution for this object.

4.23.4 Member Data Documentation

4.23.4.1 unsigned int StressTensor::_dir [protected]

Direction variable for direction this kernel acts on (0=x, 1=y, 2=z)

Definition at line 80 of file StressTensor.h.

4.23.4.2 const VariableGradient& StressTensor::_ux_grad [protected]

Velocity gradient for d(ux)/dx.

Definition at line 74 of file StressTensor.h.

4.23.4.3 const unsigned int StressTensor::_ux_var [protected]

Variable identification for ux.

Definition at line 82 of file StressTensor.h.

4.23.4.4 const VariableGradient& StressTensor::_uy_grad [protected]

Velocity gradient for d(uy)/dy.

Definition at line 75 of file StressTensor.h.

4.23.4.5 const unsigned int StressTensor::_uy_var [protected]

Variable identification for uy.

Definition at line 83 of file StressTensor.h.

4.23.4.6 const VariableGradient& StressTensor::_uz_grad [protected]

Velocity gradient for d(uz)/dz.

Definition at line 76 of file StressTensor.h.

4.23.4.7 const unsigned int StressTensor::_uz_var [protected]

Variable identification for uz.

Definition at line 84 of file StressTensor.h.

4.23.4.8 const unsigned int StressTensor::_vis_var [protected]

Variable identification for viscosity.

Definition at line 85 of file StressTensor.h.

4.23.4.9 const VariableValue& StressTensor::_viscosity [protected]

Viscosity of the fluid.

Definition at line 78 of file StressTensor.h.

The documentation for this class was generated from the following file:

- StressTensor.h

5 File Documentation

5.1 AccumulatedMaterial.h File Reference

Auxillary kernel to keep track of the total accumulated amount of a variable.

```
#include "AuxKernel.h"
```

Classes

- class AccumulatedMaterial

  AccumulatedMaterial class inherits from AuxKernel.

Functions

- template<>
  InputParameters validParams< AccumulatedMaterial > ()

5.1.1 Detailed Description

Auxillary kernel to keep track of the total accumulated amount of a variable.

This file creates an auxillary kernel that computes the total accumulated amount of a non-linear variable that has passed through a particular element in the mesh. This kernel couples with a non-linear variable and integrates it over the volume of the current element. That integrated amount is then continuously added to prior integrals calculated to create a running total of material that has passed through the element.

Author

Austin Ladshaw

Date

05/18/2018

Copyright

This kernel was designed and built at the Georgia Institute of Technology by Austin Ladshaw for PhD research in the area of radioactive particle transport and settling following a nuclear event. It was developed for the US DOD under DTRA project No. 14-24-FRCWMD-BAA. Portions Copyright (c) 2018, all rights reserved.

Austin Ladshaw does not claim any ownership or copyright to the MOOSE framework in which these kernels are constructed, only the kernels themselves. The MOOSE framework copyright is held by the Battelle Energy Alliance, LLC (c) 2010, all rights reserved.

5.1.2 Function Documentation

5.1.2.1 template< > InputParameters validParams< **AccumulatedMaterial** > ( )

5.2 ConstantEllipsoidIC.h File Reference

Initial Condition kernel for an Ellipsoid Puff of Particles.

```
#include "InitialCondition.h"
```

Classes

- class ConstantEllipsoidIC

  ConcentrationIC class object inherits from InitialCondition object.

Functions

- template< >
  InputParameters validParams< ConstantEllipsoidIC > ()

5.2.1 Detailed Description

Initial Condition kernel for an Ellipsoid Puff of Particles.

This file creates an initial condition for a non-linear variable that is dispersed in an ellipsoid pattern in space. Area inside the ellipsoid is given one value and outside the ellipsoid is given another. The ellipsoid boundary can be smoothed based on a smoothing distance that will distribute the non-linear variable linearly from the outer to the inner ellipsoid.

Author

Austin Ladshaw

Date

05/18/2018

Copyright

This kernel was designed and built at the Georgia Institute of Technology by Austin Ladshaw for PhD research in the area of radioactive particle transport and settling following a nuclear event. It was developed for the US DOD under DTRA project No. 14-24-FRCWMD-BAA. Portions Copyright (c) 2018, all rights reserved.

Austin Ladshaw does not claim any ownership or copyright to the MOOSE framework in which these kernels are constructed, only the kernels themselves. The MOOSE framework copyright is held by the Battelle Energy Alliance, LLC (c) 2010, all rights reserved.

5.2.2 Function Documentation

5.2.2.1 template<> InputParameters validParams< ConstantEllipsoidIC >( )

5.3 CoupledCoeffTimeDerivative.h File Reference

Standard kernel for coupling time derivatives.

```
#include "Kernel.h"
```

Classes

- class CoupledCoeffTimeDerivative

  CoupledCoeffTimeDerivative class object inherits from Kernel object.

Functions

- template<>
  InputParameters validParams< CoupledCoeffTimeDerivative > ()

5.3.1 Detailed Description

Standard kernel for coupling time derivatives.

This file creates a standard MOOSE kernel for the coupling of time derivative functions between different non-linear variables. It will serve as the basis for creating future heat and mass transfer kernels.

Author

Austin Ladshaw

Date

03/30/2017

Copyright

This kernel was designed and built at the Georgia Institute of Technology by Austin Ladshaw for PhD research in the area of adsorption and surface science and was developed for use by Idaho National Laboratory and Oak Ridge National Laboratory engineers and scientists. Portions Copyright (c) 2017, all rights reserved.

Austin Ladshaw does not claim any ownership or copyright to the MOOSE framework in which these kernels are constructed, only the kernels themselves. The MOOSE framework copyright is held by the Battelle Energy Alliance, LLC (c) 2010, all rights reserved.

5.3.2 Function Documentation

5.3.2.1 template<> InputParameters validParams< **CoupledCoeffTimeDerivative** > ( )

5.4 DGAdvection.h File Reference

Discontinous Galerkin kernel for advection.

```
#include "DGKernel.h"
#include <cmath>
```

Classes

- class DGAdvection

  DGAdvection class object inherits from DGKernel object.

Functions

- template<>
  InputParameters validParams< DGAdvection > ()

5.4.1 Detailed Description

Discontinous Galerkin kernel for advection.

This file creates a discontinous Galerkin kernel for advection physics in a given domain. It is a generic advection kernel that is meant to be inherited from to make a more specific kernel for a given problem. The physical parameter in this kernel's formulation is a velocity vector. That vector can be built piecewise by the respective x, y, and z components of a velocity field at a given quadrature point.

Note

> Any DG kernel under DGOSPREY will have a cooresponding G kernel (usually of same name) that must be included with the DG kernel in the input file. This is because the DG finite element method breaks into several different residual pieces, only a handful of which are handled by the DG kernel system and the other parts must be handled by the standard Galerkin system. This my be due to some legacy code in MOOSE. I am not sure if it is possible to lump all of these actions into a single DG kernel.

Author

> Austin Ladshaw

Date

> 11/20/2015

Copyright

> This kernel was designed and built at the Georgia Institute of Technology by Austin Ladshaw for PhD research in the area of adsorption and surface science and was developed for use by Idaho National Laboratory and Oak Ridge National Laboratory engineers and scientists. Portions Copyright (c) 2015, all rights reserved.

Austin Ladshaw does not claim any ownership or copyright to the MOOSE framework in which these kernels are constructed, only the kernels themselves. The MOOSE framework copyright is held by the Battelle Energy Alliance, LLC (c) 2010, all rights reserved.

5.4.2 Function Documentation

5.4.2.1 template<> InputParameters validParams< DGAdvection >( )

5.5 DGAnisotropicDiffusion.h File Reference

Discontinous Galerkin kernel for anisotropic diffusion.

```
#include "DGKernel.h"
#include "MooseVariable.h"
#include <cmath>
```

Classes

- class DGAnisotropicDiffusion

  DGAnisotropicDiffusion class object inherits from DGKernel object.

Functions

- template<>
  InputParameters validParams< DGAnisotropicDiffusion > ()

5.5.1 Detailed Description

Discontinous Galerkin kernel for anisotropic diffusion.

This file creates a discontinous Galerkin kernel for anisotropic diffusion in a given domain. It is a generic diffusion kernel that is meant to be inherited from to make a more specific kernel for a given problem. The physical parameter in this kernel's formulation is a diffusion tensor. That tensor can be built piecewise by the respective components of the tensor at a given quadrature point.

Note

> Any DG kernel under DGOSPREY will have a cooresponding G kernel (usually of same name) that must be included with the DG kernel in the input file. This is because the DG finite element method breaks into several different residual pieces, only a handful of which are handled by the DG kernel system and the other parts must be handled by the standard Galerkin system. This my be due to some legacy code in MOOSE. I am not sure if it is possible to lump all of these actions into a single DG kernel.

Author

> Austin Ladshaw

Date

> 11/20/2015

Copyright

> This kernel was designed and built at the Georgia Institute of Technology by Austin Ladshaw for PhD research in the area of adsorption and surface science and was developed for use by Idaho National Laboratory and Oak Ridge National Laboratory engineers and scientists. Portions Copyright (c) 2015, all rights reserved.

Austin Ladshaw does not claim any ownership or copyright to the MOOSE framework in which these kernels are constructed, only the kernels themselves. The MOOSE framework copyright is held by the Battelle Energy Alliance, LLC (c) 2010, all rights reserved.

5.5.2 Function Documentation

5.5.2.1 template<> InputParameters validParams< **DGAnisotropicDiffusion** > ( )

5.6 DGConcentrationAdvection.h File Reference

Discontinous Galerkin kernel for density advection.

```
#include "DGAdvection.h"
```

Classes

- class DGConcentrationAdvection

  DGConcentrationAdvection class object inherits from DGKernel object.

Functions

- template<>
  InputParameters validParams< DGConcentrationAdvection > ()

5.6.1 Detailed Description

Discontinous Galerkin kernel for density advection.

This file creates a discontinous Galerkin kernel for density advection in a given domain. It is a generic advection kernel that is meant to be inherited from to make a more specific kernel for a given problem.

Note

Any DG kernel under FENNEC will have a cooresponding G kernel (usually of same name) that must be included with the DG kernel in the input file. This is because the DG finite element method breaks into several different residual pieces, only a handful of which are handled by the DG kernel system and the other parts must be handled by the standard Galerkin system. This my be due to some legacy code in MOOSE. I am not sure if it is possible to lump all of these actions into a single DG kernel.

Author

Austin Ladshaw

Date

07/12/2018

Copyright

This kernel was designed and built at the Georgia Institute of Technology by Austin Ladshaw for PhD research in the area of radioactive particle transport and settling following a nuclear event. It was developed for the US DOD under DTRA project No. 14-24-FRCWMD-BAA. Portions Copyright (c) 2018, all rights reserved.

Austin Ladshaw does not claim any ownership or copyright to the MOOSE framework in which these kernels are constructed, only the kernels themselves. The MOOSE framework copyright is held by the Battelle Energy Alliance, LLC (c) 2010, all rights reserved.

5.6.2 Function Documentation

5.6.2.1 template<> InputParameters validParams< DGConcentrationAdvection > ( )

5.7 DGConcentrationFluxBC.h File Reference

Boundary Condition kernel for the flux of concentration/density across a boundary of the domain.

```
#include "DGFluxBC.h"
```

Classes

- class DGConcentrationFluxBC

  DGConcentrationFluxBC class object inherits from IntegratedBC object.

Functions

- template<>
  InputParameters validParams< DGConcentrationFluxBC > ()

5.7.1 Detailed Description

Boundary Condition kernel for the flux of concentration/density across a boundary of the domain.

This file creates a generic boundary condition kernel for the flux of matter accross a boundary. The flux is based on a velocity vector and is valid in all directions and all boundaries of a DG method. Since the DG method's flux boundary conditions are essitially the same for input and ouput boundaries, this kernel will check the sign of the flux normal to the boundary and determine automattically whether it is an output or input boundary, then apply the appropriate conditions.

Author

Austin Ladshaw

Date

07/12/2018

Copyright

This kernel was designed and built at the Georgia Institute of Technology by Austin Ladshaw for PhD research in the area of radioactive particle transport and settling following a nuclear event. It was developed for the US DOD under DTRA project No. 14-24-FRCWMD-BAA. Portions Copyright (c) 2018, all rights reserved.

Austin Ladshaw does not claim any ownership or copyright to the MOOSE framework in which these kernels are constructed, only the kernels themselves. The MOOSE framework copyright is held by the Battelle Energy Alliance, LLC (c) 2010, all rights reserved.

5.7.2 Function Documentation

5.7.2.1 template<> InputParameters validParams< DGConcentrationFluxBC > ( )

5.8 DGContinuumBC.h File Reference

Boundary Condition kernel for the continuity equation applied at boundary conditions.

```
#include "DGMomentumFluxBC.h"
```

Classes

- class DGContinuumBC

  DGMomentumFluxBC class object inherits from IntegratedBC object.

Functions

- template<>
  InputParameters validParams< DGContinuumBC > ()

5.8.1 Detailed Description

Boundary Condition kernel for the continuity equation applied at boundary conditions.

This file creates a boundary condition kernel for the continuity of momemtum at boundaries. Use this BC for systems involving the conservation of momentum where no additional momentum is being added to the system from the boundaries.

Author

Austin Ladshaw

Date

07/10/2018

Copyright

This kernel was designed and built at the Georgia Institute of Technology by Austin Ladshaw for PhD research in the area of radioactive particle transport and settling following a nuclear event. It was developed for the US DOD under DTRA project No. 14-24-FRCWMD-BAA. Portions Copyright (c) 2018, all rights reserved.

Austin Ladshaw does not claim any ownership or copyright to the MOOSE framework in which these kernels are constructed, only the kernels themselves. The MOOSE framework copyright is held by the Battelle Energy Alliance, LLC (c) 2010, all rights reserved.

5.8.2 Function Documentation

5.8.2.1 template<> InputParameters validParams< DGContinuumBC > ( )

5.9 DGFluxBC.h File Reference

Boundary Condition kernel for the flux across a boundary of the domain.

```
#include "IntegratedBC.h"
#include "libmesh/vector_value.h"
```

Classes

- class DGFluxBC

  DGFluxBC class object inherits from IntegratedBC object.

Functions

- template<>
  InputParameters validParams< DGFluxBC > ()

5.9.1 Detailed Description

Boundary Condition kernel for the flux across a boundary of the domain.

This file creates a generic boundary condition kernel for the flux of material accross a boundary. The flux is based on a velocity vector and is valid in all directions and all boundaries of a DG method. Since the DG method's flux boundary conditions are essitially the same for input and ouput boundaries, this kernel will check the sign of the flux normal to the boundary and determine automattically whether it is an output or input boundary, then apply the appropriate conditions.

This type of boundary condition for DG kernels applies the true flux boundary condition. Alternatively, you can use the "FluxLimitedBC" to impose a Dirichlet boundary condition on the system. Although, in true finite volumes or DG methods, there is no Dirichlet boundary conditions, because the solutions are based on fluxes into and out of cells in a domain.

Author

Austin Ladshaw

Date

11/20/2015

Copyright

This kernel was designed and built at the Georgia Institute of Technology by Austin Ladshaw for PhD research in the area of adsorption and surface science and was developed for use by Idaho National Laboratory and Oak Ridge National Laboratory engineers and scientists. Portions Copyright (c) 2015, all rights reserved.

Austin Ladshaw does not claim any ownership or copyright to the MOOSE framework in which these kernels are constructed, only the kernels themselves. The MOOSE framework copyright is held by the Battelle Energy Alliance, LLC (c) 2010, all rights reserved.

5.9.2 Function Documentation

5.9.2.1 template<> InputParameters validParams< DGFluxBC > ( )

5.10 DGFluxLimitedBC.h File Reference

Boundary Condition kernel to mimic a Dirichlet BC for DG methods.

```
#include "IntegratedBC.h"
#include "libmesh/vector_value.h"
#include "MooseVariable.h"
```

Classes

- class DGFluxLimitedBC

  DGFluxLimitedBC class object inherits from IntegratedBC object.

Functions

- template<>
  InputParameters validParams< DGFluxLimitedBC > ()

5.10.1 Detailed Description

Boundary Condition kernel to mimic a Dirichlet BC for DG methods.

This file creates a boundary condition kernel to impose a dirichlet-like boundary condition in DG methods. True DG methods do not have Dirichlet boundary conditions, so this kernel seeks to impose a constraint on the inlet of a boundary that is met if the value of a variable at the inlet boundary is equal to the finite element solution at that boundary. When the condition is not met, the residuals get penalized until the condition is met.

Author

Austin Ladshaw

Date

11/20/2015

Copyright

This kernel was designed and built at the Georgia Institute of Technology by Austin Ladshaw for PhD research in the area of adsorption and surface science and was developed for use by Idaho National Laboratory and Oak Ridge National Laboratory engineers and scientists. Portions Copyright (c) 2015, all rights reserved.

Austin Ladshaw does not claim any ownership or copyright to the MOOSE framework in which these kernels are constructed, only the kernels themselves. The MOOSE framework copyright is held by the Battelle Energy Alliance, LLC (c) 2010, all rights reserved.

5.10.2 Function Documentation

5.10.2.1 template<> InputParameters validParams< DGFluxLimitedBC > ( )

5.11 DGMomentumAdvection.h File Reference

Discontinous Galerkin kernel for momentum advection.

```
#include "DGAdvection.h"
```

Classes

- class DGMomentumAdvection

  DGMomentumAdvection class object inherits from DGKernel object.

Functions

- template<>
  InputParameters validParams< DGMomentumAdvection > ()

5.11.1 Detailed Description

Discontinous Galerkin kernel for momentum advection.

This file creates a discontinous Galerkin kernel for momentum advection in a given domain. It is a generic advection kernel that is meant to be inherited from to make a more specific kernel for a given problem.

Note

Any DG kernel under FENNEC will have a cooresponding G kernel (usually of same name) that must be included with the DG kernel in the input file. This is because the DG finite element method breaks into several different residual pieces, only a handful of which are handled by the DG kernel system and the other parts must be handled by the standard Galerkin system. This my be due to some legacy code in MOOSE. I am not sure if it is possible to lump all of these actions into a single DG kernel.

Author

Austin Ladshaw

Date

07/09/2018

Copyright

This kernel was designed and built at the Georgia Institute of Technology by Austin Ladshaw for PhD research in the area of radioactive particle transport and settling following a nuclear event. It was developed for the US DOD under DTRA project No. 14-24-FRCWMD-BAA. Portions Copyright (c) 2018, all rights reserved.

Austin Ladshaw does not claim any ownership or copyright to the MOOSE framework in which these kernels are constructed, only the kernels themselves. The MOOSE framework copyright is held by the Battelle Energy Alliance, LLC (c) 2010, all rights reserved.

5.11.2 Function Documentation

5.11.2.1 template<> InputParameters validParams< **DGMomentumAdvection** > ( )

5.12 DGMomentumDiffusion.h File Reference

Discontinous Galerkin kernel for viscous momentum dispersion.

```
#include "DGAnisotropicDiffusion.h"
```

Classes

- class DGMomentumDiffusion

  DGMomentumDiffusion class object inherits from DGKernel object.

Functions

- template<>
  InputParameters validParams< DGMomentumDiffusion > ()

5.12.1 Detailed Description

Discontinous Galerkin kernel for viscous momentum dispersion.

This file creates a discontinous Galerkin kernel for momentum diffusion. It is build on utilizing the existing DG←AnisotropicDiffusion kernel and replaces the Diffusion coefficient with fluid viscosity.

Note

> Any DG kernel under FENNEC will have a cooresponding G kernel (usually of same name) that must be included with the DG kernel in the input file. This is because the DG finite element method breaks into several different residual pieces, only a handful of which are handled by the DG kernel system and the other parts must be handled by the standard Galerkin system. This my be due to some legacy code in MOOSE. I am not sure if it is possible to lump all of these actions into a single DG kernel.

Author

> Austin Ladshaw

Date

> 07/09/2018

Copyright

> This kernel was designed and built at the Georgia Institute of Technology by Austin Ladshaw for PhD research in the area of radioactive particle transport and settling following a nuclear event. It was developed for the US DOD under DTRA project No. 14-24-FRCWMD-BAA. Portions Copyright (c) 2018, all rights reserved.

Austin Ladshaw does not claim any ownership or copyright to the MOOSE framework in which these kernels are constructed, only the kernels themselves. The MOOSE framework copyright is held by the Battelle Energy Alliance, LLC (c) 2010, all rights reserved.

5.12.2 Function Documentation

5.12.2.1 template<> InputParameters validParams< DGMomentumDiffusion > ( )

5.13 DGMomentumFluxBC.h File Reference

Boundary Condition kernel for the flux of momentum across a boundary of the domain.

```
#include "DGFluxBC.h"
```

Classes

- class DGMomentumFluxBC

  DGMomentumFluxBC class object inherits from IntegratedBC object.

Functions

- template<>
  InputParameters validParams< DGMomentumFluxBC > ()

5.13.1 Detailed Description

Boundary Condition kernel for the flux of momentum across a boundary of the domain.

This file creates a generic boundary condition kernel for the flux of momentum accross a boundary. The flux is based on a velocity vector and is valid in all directions and all boundaries of a DG method. Since the DG method's flux boundary conditions are essitially the same for input and ouput boundaries, this kernel will check the sign of the flux normal to the boundary and determine automattically whether it is an output or input boundary, then apply the appropriate conditions.

Author

Austin Ladshaw

Date

07/09/2018

Copyright

This kernel was designed and built at the Georgia Institute of Technology by Austin Ladshaw for PhD research in the area of radioactive particle transport and settling following a nuclear event. It was developed for the US DOD under DTRA project No. 14-24-FRCWMD-BAA. Portions Copyright (c) 2018, all rights reserved.

Austin Ladshaw does not claim any ownership or copyright to the MOOSE framework in which these kernels are constructed, only the kernels themselves. The MOOSE framework copyright is held by the Battelle Energy Alliance, LLC (c) 2010, all rights reserved.

5.13.2 Function Documentation

5.13.2.1 template<> InputParameters validParams< DGMomentumFluxBC >( )

5.14 fennecApp.h File Reference

```
#include "MooseApp.h"
```

Classes

- class fennecApp

Functions

- template<>
  InputParameters validParams< fennecApp > ()

5.14.1 Function Documentation

5.14.1.1 template<> InputParameters validParams< fennecApp > ( )

5.15 GAdvection.h File Reference

Kernel for use with the corresponding DGAdvection object.

```
#include "Kernel.h"
```

Classes

- class GAdvection

  GAdvection class object inherits from Kernel object.

Functions

- template<>
  InputParameters validParams< GAdvection > ()

5.15.1 Detailed Description

Kernel for use with the corresponding DGAdvection object.

This file creates a standard MOOSE kernel that is to be used in conjunction with the DGAdvection kernel for the discontinous Galerkin formulation of advection physics in MOOSE. In order to complete the DG formulation of the advective physics, this kernel must be utilized with every variable that also uses the DGAdvection kernel.

Author

Austin Ladshaw

Date

11/20/2015

Copyright

This kernel was designed and built at the Georgia Institute of Technology by Austin Ladshaw for PhD research in the area of adsorption and surface science and was developed for use by Idaho National Laboratory and Oak Ridge National Laboratory engineers and scientists. Portions Copyright (c) 2015, all rights reserved.

Austin Ladshaw does not claim any ownership or copyright to the MOOSE framework in which these kernels are constructed, only the kernels themselves. The MOOSE framework copyright is held by the Battelle Energy Alliance, LLC (c) 2010, all rights reserved.

5.15.2 Function Documentation

5.15.2.1 template<> InputParameters validParams< GAdvection >( )

5.16 GAnisotropicDiffusion.h File Reference

Kernel for use with the corresponding DGAnisotropicDiffusion object.

```
#include "Kernel.h"
```

Classes

- class GAnisotropicDiffusion

  GAnisotropicDiffusion class object inherits from Kernel object.

Functions

- template<>
  InputParameters validParams< GAnisotropicDiffusion > ()

5.16.1 Detailed Description

Kernel for use with the corresponding DGAnisotropicDiffusion object.

This file creates a standard MOOSE kernel that is to be used in conjunction with the DGAnisotropicDiffusion kernel for the discontinous Galerkin formulation of advection physics in MOOSE. In order to complete the DG formulation of the advective physics, this kernel must be utilized with every variable that also uses the DGAAnisotropicDiffusion kernel.

Author

Austin Ladshaw

Date

11/20/2015

Copyright

This kernel was designed and built at the Georgia Institute of Technology by Austin Ladshaw for PhD research in the area of adsorption and surface science and was developed for use by Idaho National Laboratory and Oak Ridge National Laboratory engineers and scientists. Portions Copyright (c) 2015, all rights reserved.

Austin Ladshaw does not claim any ownership or copyright to the MOOSE framework in which these kernels are constructed, only the kernels themselves. The MOOSE framework copyright is held by the Battelle Energy Alliance, LLC (c) 2010, all rights reserved.

5.16.2 Function Documentation

5.16.2.1 template<> InputParameters validParams< GAnisotropicDiffusion > ( )

5.17 GConcentrationAdvection.h File Reference

Kernel for use with the corresponding DGConcentrationAdvection object.

```
#include "GAdvection.h"
```

Classes

- class GConcentrationAdvection

  GConcentrationAdvection class object inherits from Kernel object.

Functions

- template<>
  InputParameters validParams< GConcentrationAdvection > ()

5.17.1 Detailed Description

Kernel for use with the corresponding DGConcentrationAdvection object.

This file creates a standard MOOSE kernel that is to be used in conjunction with DGConcentrationAdvection for the discontinous Galerkin formulation of momentum advection in MOOSE. In order to complete the DG formulation of the advective physics, this kernel must be utilized with every variable that also uses the DGConcentrationAdvection kernel.

Author

Austin Ladshaw

Date

07/12/2018

Copyright

This kernel was designed and built at the Georgia Institute of Technology by Austin Ladshaw for PhD research in the area of radioactive particle transport and settling following a nuclear event. It was developed for the US DOD under DTRA project No. 14-24-FRCWMD-BAA. Portions Copyright (c) 2018, all rights reserved.

Austin Ladshaw does not claim any ownership or copyright to the MOOSE framework in which these kernels are constructed, only the kernels themselves. The MOOSE framework copyright is held by the Battelle Energy Alliance, LLC (c) 2010, all rights reserved.

5.17.2 Function Documentation

5.17.2.1 template<> InputParameters validParams< GConcentrationAdvection > ( )

5.18 GMomentumAdvection.h File Reference

Kernel for use with the corresponding DGMomentumAdvection object.

```
#include "GAdvection.h"
```

Classes

- class GMomentumAdvection

  GAdvection class object inherits from Kernel object.

Functions

- template<>
  InputParameters validParams< GMomentumAdvection > ()

5.18.1 Detailed Description

Kernel for use with the corresponding DGMomentumAdvection object.

This file creates a standard MOOSE kernel that is to be used in conjunction with DGMomentumAdvection for the discontinous Galerkin formulation of momentum advection in MOOSE. In order to complete the DG formulation of the advective physics, this kernel must be utilized with every variable that also uses the DGMomentumAdvection kernel.

Author

Austin Ladshaw

Date

07/09/2018

Copyright

This kernel was designed and built at the Georgia Institute of Technology by Austin Ladshaw for PhD research in the area of radioactive particle transport and settling following a nuclear event. It was developed for the US DOD under DTRA project No. 14-24-FRCWMD-BAA. Portions Copyright (c) 2018, all rights reserved.

Austin Ladshaw does not claim any ownership or copyright to the MOOSE framework in which these kernels are constructed, only the kernels themselves. The MOOSE framework copyright is held by the Battelle Energy Alliance, LLC (c) 2010, all rights reserved.

5.18.2 Function Documentation

5.18.2.1 template<> InputParameters validParams< **GMomentumAdvection** > ( )

5.19 GMomentumDiffusion.h File Reference

Kernel for use with the corresponding DGMomentumDiffusion object.

```
#include "GAnisotropicDiffusion.h"
```

Classes

- class GMomentumDiffusion

  GAnisotropicDiffusion class object inherits from Kernel object.

Functions

- template<>
  InputParameters validParams< GMomentumDiffusion > ()

5.19.1 Detailed Description

Kernel for use with the corresponding DGMomentumDiffusion object.

This file creates a standard MOOSE kernel that is to be used in conjunction with the DGMomentumDiffusion kernel for the discontinous Galerkin formulation of momentum conservation in MOOSE.

Author

Austin Ladshaw

Date

07/09/2018

Copyright

This kernel was designed and built at the Georgia Institute of Technology by Austin Ladshaw for PhD research in the area of radioactive particle transport and settling following a nuclear event. It was developed for the US DOD under DTRA project No. 14-24-FRCWMD-BAA. Portions Copyright (c) 2018, all rights reserved.

Austin Ladshaw does not claim any ownership or copyright to the MOOSE framework in which these kernels are constructed, only the kernels themselves. The MOOSE framework copyright is held by the Battelle Energy Alliance, LLC (c) 2010, all rights reserved.

5.19.2 Function Documentation

5.19.2.1 template<> InputParameters validParams< GMomentumDiffusion > ( )

5.20 MomentumAcceleration.h File Reference

Kernel to couple density and acceleration variables to a momentum conservation equation.

```
#include "Kernel.h"
```

Classes

- class MomentumAcceleration

  MomentumAcceleration class object inherits from Kernel object.

Functions

- template<>
  InputParameters validParams< MomentumAcceleration > ()

5.20.1 Detailed Description

Kernel to couple density and acceleration variables to a momentum conservation equation.

This file creates a standard MOOSE kernel that is to be used to coupled acceleration variables (such as gravity) with density variables within the momentum balance for fluid velocity in a particular direction (x, y, z). This kernel will act on a velocity variable in x, y, or z, but not explicitly use that variable. So, as a result there is no Jacobian for this residual. Instead, there will be two off diagonal Jacobians, which indicate the coupling of density and acceleration variables.

Author

Austin Ladshaw

Date

07/12/2018

Copyright

This kernel was designed and built at the Georgia Institute of Technology by Austin Ladshaw for PhD research in the area of radioactive particle transport and settling following a nuclear event. It was developed for the US DOD under DTRA project No. 14-24-FRCWMD-BAA. Portions Copyright (c) 2018, all rights reserved.

Austin Ladshaw does not claim any ownership or copyright to the MOOSE framework in which these kernels are constructed, only the kernels themselves. The MOOSE framework copyright is held by the Battelle Energy Alliance, LLC (c) 2010, all rights reserved.

5.20.2 Function Documentation

5.20.2.1 template<> InputParameters validParams< **MomentumAcceleration** > ( )

5.21 MomentumAccumulation.h File Reference

Time Derivative kernel for the accumulation of momentum of a component of velocity.

```
#include "TimeDerivative.h"
```

Classes

- class MomentumAccumulation

    MomentumAccumulation class object inherits from TimeDerivative object.

Functions

- template<>
  InputParameters validParams< MomentumAccumulation > ()

5.21.1 Detailed Description

Time Derivative kernel for the accumulation of momentum of a component of velocity.

This file creates a time derivative kernel to be used in the momentum transport equations for a velocity component.

Author

Austin Ladshaw

Date

07/09/2018

Copyright

This kernel was designed and built at the Georgia Institute of Technology by Austin Ladshaw for PhD research in the area of radioactive particle transport and settling following a nuclear event. It was developed for the US DOD under DTRA project No. 14-24-FRCWMD-BAA. Portions Copyright (c) 2018, all rights reserved.

Austin Ladshaw does not claim any ownership or copyright to the MOOSE framework in which these kernels are constructed, only the kernels themselves. The MOOSE framework copyright is held by the Battelle Energy Alliance, LLC (c) 2010, all rights reserved.

5.21.2 Function Documentation

5.21.2.1 template<> InputParameters validParams< MomentumAccumulation > ( )

5.22 MomentumPressureGrad.h File Reference

Kernel to couple a pressure gradient to a momentum conservation equation.

```
#include "Kernel.h"
```

Classes

- class MomentumPressureGrad

  MomentumPressureGrad class object inherits from Kernel object.

Functions

- template<>
  InputParameters validParams< MomentumPressureGrad > ()

5.22.1 Detailed Description

Kernel to couple a pressure gradient to a momentum conservation equation.

This file creates a standard MOOSE kernel that is to be used to coupled a pressure gradient to the conservation of momentum equation for fluid flow in a particular direction (x, y, z). The direction on which this kernel acts must be explicitly given to know which component of the pressure gradient to use. Also, this kernel will act on a velocity variable in that direction, but not use that variable directly. Thus, the Jacobian will return zero, but the off-diagonal Jacobian will be non-zero.

Author

Austin Ladshaw

Date

07/12/2018

Copyright

This kernel was designed and built at the Georgia Institute of Technology by Austin Ladshaw for PhD research in the area of radioactive particle transport and settling following a nuclear event. It was developed for the US DOD under DTRA project No. 14-24-FRCWMD-BAA. Portions Copyright (c) 2018, all rights reserved.

Austin Ladshaw does not claim any ownership or copyright to the MOOSE framework in which these kernels are constructed, only the kernels themselves. The MOOSE framework copyright is held by the Battelle Energy Alliance, LLC (c) 2010, all rights reserved.

5.22.2 Function Documentation

5.22.2.1 template<> InputParameters validParams< **MomentumPressureGrad** > ( )

5.23 StressTensor.h File Reference

Kernel used to integrate a Stress Tensor into conservation of momentum.

```
#include "Kernel.h"
```

Classes

- class StressTensor

  StressTensor class object inherits from Kernel object.

Functions

- template<>
  InputParameters validParams< StressTensor > ()

5.23.1 Detailed Description

Kernel used to integrate a Stress Tensor into conservation of momentum.

This file creates a standard MOOSE kernel that is to be used in with other conservation of momentum kernels. Momentum conservation couples velocity terms together in Cartesian coordinates.

Author

Austin Ladshaw

Date

07/09/2018

Copyright

This kernel was designed and built at the Georgia Institute of Technology by Austin Ladshaw for PhD research in the area of radioactive particle transport and settling following a nuclear event. It was developed for the US DOD under DTRA project No. 14-24-FRCWMD-BAA. Portions Copyright (c) 2018, all rights reserved.

Austin Ladshaw does not claim any ownership or copyright to the MOOSE framework in which these kernels are constructed, only the kernels themselves. The MOOSE framework copyright is held by the Battelle Energy Alliance, LLC (c) 2010, all rights reserved.

5.23.2 Function Documentation

5.23.2.1 template<> InputParameters validParams< **StressTensor** > ( )

Index