Equation Forms

Coefficient Form PDE

$$\begin{cases} e_a\dfrac{\partial^2 u}{\partial t^2} + d_a\dfrac{\partial u}{\partial t} + \nabla\cdot(-c\nabla u - \alpha u + \gamma) + \beta\cdot\nabla u + au = f & \text{in } \Omega \\ \mathbf{n}\cdot(c\nabla u + \alpha u - \gamma) + qu = g - h^T\mu & \text{on } \partial\Omega \\ hu = r & \text{on } \partial\Omega \end{cases}$$

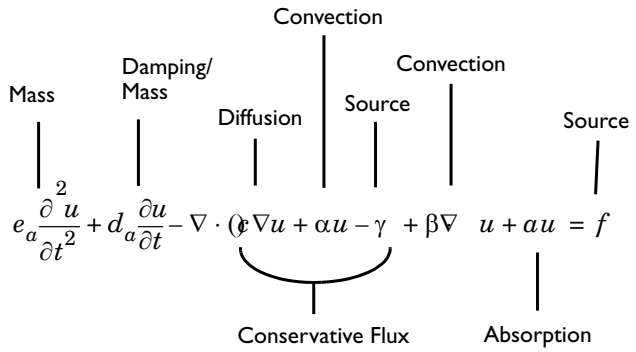
General Form PDE

$$\begin{cases} \nabla\cdot\Gamma = F & \text{in } \Omega \\ -\mathbf{n}\cdot\Gamma = G + \left(\dfrac{\partial R}{\partial u}\right)^T\mu & \text{on } \partial\Omega \\ 0 = R & \text{on } \partial\Omega \end{cases}$$

Interpreting PDE Coefficients

The COMSOL Multiphysics PDE formulations can model a variety of problems, but note that this documentation uses coefficient names that fall within the realm of continuum mechanics and mass transfer. For the coefficient form:

- e_a is the *mass coefficient*
- d_a is a *damping coefficient* or a *mass coefficient*.
- c is the *diffusion coefficient*.
- α is the *conservative flux convection coefficient*.
- β is the *convection coefficient*.
- a is the *absorption coefficient*.
- γ is the *conservative flux source term*.
- f is the *source term*.



In some cases, this interpretation does not apply. For instance, a time-harmonic PDE such as the Helmholtz equation represents a time-dependent phenomenon transformed into the frequency domain.

For the Neumann boundary condition of the coefficient form

$$\mathbf{n}\cdot(c\nabla u + \alpha u - \gamma) + qu = g - h^T\mu$$

- q is the *boundary absorption coefficient*.
- g is the *boundary source term*.

Classical PDEs

Many classical PDEs are instances of the coefficient form. All the classical PDEs in this section have their own application modes. To find them, go to the **Model Navigator** and then to the list of application modes; within the **PDE Modes** section find the **Classical PDEs** folder. The table below shows the available classical PDEs using two notations:

the compact notation of vector analysis (used in this documentation) and an expanded mathematical notation.

TABLE 3-1: CLASSICAL PDES IN COMPACT AND STANDARD NOTATION

EQUATION	COMPACT NOTATION	STANDARD NOTATION (2D)
Laplace's equation	$-\nabla \cdot (\nabla u) = 0$	$-\frac{\partial}{\partial x}\frac{\partial u}{\partial x} - \frac{\partial}{\partial y}\frac{\partial u}{\partial y} = 0$
Poisson's equation	$-\nabla \cdot (c\nabla u) = f$	$-\frac{\partial}{\partial x}\left(c\frac{\partial u}{\partial x}\right) - \frac{\partial}{\partial y}\left(c\frac{\partial u}{\partial y}\right) = f$
Helmholtz equation	$-\nabla \cdot (c\nabla u) + au = f$	$-\frac{\partial}{\partial x}\left(c\frac{\partial u}{\partial x}\right) - \frac{\partial}{\partial y}\left(c\frac{\partial u}{\partial y}\right) + au = f$
Heat equation	$d_a\frac{\partial u}{\partial t} - \nabla \cdot (c\nabla u) = f$	$d_a\frac{\partial u}{\partial t} - \frac{\partial}{\partial x}\left(c\frac{\partial u}{\partial x}\right) - \frac{\partial}{\partial y}\left(c\frac{\partial u}{\partial y}\right) = f$
Wave equation	$e_a\frac{\partial^2 u}{\partial t^2} - \nabla \cdot (c\nabla u) = f$	$e_a\frac{\partial^2 u}{\partial t^2} - \frac{\partial}{\partial x}\left(c\frac{\partial u}{\partial x}\right) - \frac{\partial}{\partial y}\left(c\frac{\partial u}{\partial y}\right) = f$
Schrödinger equation	$-\nabla \cdot (c\nabla u) + au = \lambda u$	$-\frac{\partial}{\partial x}\left(c\frac{\partial u}{\partial x}\right) - \frac{\partial}{\partial y}\left(c\frac{\partial u}{\partial y}\right) + au = \lambda u$
Convection-diffusion equation	$d_a\frac{\partial u}{\partial t} - \nabla \cdot (c\nabla u) + \beta \cdot \nabla u = f$	$d_a\frac{\partial u}{\partial t} - \frac{\partial}{\partial x}\left(c\frac{\partial u}{\partial x}\right) - \frac{\partial}{\partial y}\left(c\frac{\partial u}{\partial y}\right) + \beta_x\frac{\partial u}{\partial x} + \beta_y\frac{\partial u}{\partial y} = f$

Mathematical and Logical Functions

The following lists include mathematical and logical functions and operators:

TABLE 3-2: UNARY OPERATORS

OPERATOR	DESCRIPTION
+	unary plus
-	unary minus
~	logical not

TABLE 3-3: BINARY OPERATORS

OPERATOR	DESCRIPTION
+	plus
-	minus
*	multiply
/	divide
^	power
==	equal
~=	not equal
>	greater than
>=	greater than or equal to
<	less than
<=	less than or equal to
|	or
&	and

TABLE 3-4: MATHEMATICAL FUNCTIONS AND OPERATORS

FUNCTION	DESCRIPTION	SYNTAX EXAMPLE
abs	absolute value	`abs(x)`
acos	inverse cosine	`acos(x)`
acosh	inverse hyperbolic cosine	`acosh(x)`
acot	inverse cotangent	`acot(x)`
acoth	inverse hyperbolic cotangent	`acoth(x)`

FUNCTION	DESCRIPTION	SYNTAX EXAMPLE
acsc	inverse cosecant	`acsc(x)`
`acsch`	inverse hyperbolic cosecant	`acsch(x)`
`angle`	phase angle	`angle(x)`
`asec`	inverse secant	`asec(x)`
`asech`	inverse hyperbolic secant	`asech(x)`
`asin`	inverse sine	`asin(x)`
`asinh`	inverse hyperbolic sine	`asinh(x)`
`atan`	inverse tangent	`atan(x)`
`atan2`	four-quadrant inverse tangent	`atan2(y,x)`
`atanh`	inverse hyperbolic tangent	`atanh(x)`
`besselj`	Bessel function of the first kind	`besselj(a,x)`
`bessely`	Bessel function of the second kind	`bessely(a,x)`
`besseli`	Modified Bessel function of the first kind	`besseli(a,x)`
`besselk`	Modified Bessel function of the second kind	`besselk(a,x)`
`conj`	complex conjugate	`conj(x)`
`cos`	cosine	`cos(x)`
`cosh`	hyperbolic cosine	`cosh(x)`
`cot`	cotangent	`cot(x)`
`coth`	hyperbolic cotangent	`coth(x)`
`csc`	cosecant	`csc(x)`
`csch`	hyperbolic cosecant	`csch(x)`
`eps`	floating point relative accuracy	`eps`
`exp`	exponential	`exp(x)`
`flc1hs`	smoothed Heaviside function	`flc1hs(x,scale)`
`flc2hs`	smoothed Heaviside function	`flc2hs(x,scale)`
`flsmhs`	smoothed Heaviside function	`flsmhs(x,scale)`
`flsmsign`	smoothed sign function	`flsmsign(x,scale)`
`i, j`	imaginary unit	`i`
`imag`	imaginary part	`imag(u)`
`inf`	infinity	`inf`
`log`	natural logarithm	`log(x)`
`log10`	common logarithm (base 10)	`log10(x)`

FUNCTION	DESCRIPTION	SYNTAX EXAMPLE
log2	base 2 logarithm	log2(x)
max	maximum of two arguments	max(a,b)
min	minimum of two arguments	min(a,b)
mod	modulo operator	mod(a,b)
NaN, nan	not-a-number	nan
real	real part	real(u)
pi	pi	pi
sec	secant	sec(x)
sech	hyperbolic secant	sech(x)
sign	sign function	sign(u)
sin	sine	sin(x)
sinh	hyperbolic sine	sinh(x)
sqrt	square root	sqrt(x)
tan	tangent	tan(x)
tanh	hyperbolic tangent	tanh(x)

TABLE 3-5: VECTOR-CREATION FUNCTIONS AND OPERATORS

FUNCTION	DESCRIPTION
linspace(*start*, *stop*, *N*)	linearly spaced vector
logspace(*start*, *stop*, *N*)	logarithmically spaced vector
:	vector creation operator

The following modeling features support vector-valued expressions:

- Extra grid lines in the **Axes/Grid Settings** dialog box
- Line and point coordinates when using the **Line** and **Point** dialog boxes
- The times for output from the time-dependent solver and the list of parameter values for the parametric solvers in the **Solver Parameter** dialog box
- The contour levels, the streamline start point coordinates, and the coordinates in arrow plots. These visualization settings appear in the **Plot Parameters** dialog box.
- The edge vertex distribution on boundary segments in the **Mapped Mesh Parameters** dialog box.
- The element layer distribution in the **Extrude Mesh** and **Revolve Mesh** dialog boxes.

Variables

Geometry Variables

In the tables below, *x* (in italic font) indicates the name of any space coordinate in the current model, for example, *x*, *y*, and *z*. *u* indicates the name of a dependent variable. Replace it with the actual names of the dependent variables in your model, for example, *T* for temperature.

TABLE 3-6: GEOMETRY VARIABLES

VARIABLE NAME	DESCRIPTION
`x y z`	Default space coordinate names, Cartesian coordinates
`r phi z`	Default space coordinate names, cylindrical coordinates
x`g`	Space coordinate values of the original geometry
`s`	Curve parameter in 2D (0 to 1 in direction of the boundary arrow)
`s1 s2`	Arc length parameters in 3D
`t`*x*	Curve tangent vector, *x* component (2D)
`t1`*x* `t2`*x*	Surface tangent vectors, *x* component (3D)
`n`*x*	Outward unit normal vector, *x* component
`dn`*x*	Down direction normal vector, *x* component
`un`*x*	Up direction normal vector, *x* component
`h`	Mesh element diameter
`dom`	Domain number
`dvol`	Determinant of the Jacobian relating local space coordinate values to the global coordinate values

Field Variables

Table 3-7 summarizes the field variables that are available in all COMSOL Multiphysics models. The table does not include *application mode variables*, which vary depending on the application modes in your model. See the *COMSOL Multiphysics Modeling Guide* and the module documentation for more information about available application mode variables.

TABLE 3-7: FIELD VARIABLES

VARIABLE NAME	DESCRIPTION
u	Dependent variable
ux	Dependent variable, space derivative w.r.t x
ux_ix_j	Dependent variable, second space derivative w.r.t x_i and x_j
u`t`	Dependent variable, first time derivative
u`tt`	Dependent variable, second time derivative
ux_i`t`	Dependent variable, mixed space and first time derivative
ux_i`tt`	Dependent variable, mixed space and second time derivative
uTx	Tangential derivative variable

Miscellaneous Variables

TABLE 3-8: MISCELLANEOUS VARIABLES

VARIABLE NAME	DESCRIPTION
`t`	time
`lambda`	eigenvalue (for postprocessing only)
`phase`	phase factor

Operators

TABLE 3-9: OPERATORS

OPERATOR	DESCRIPTION
`diff(`*`f`*`, `*`x`*`)`	Differentiation operator. Differentiation of *`f`* with respect to *`x`*
`pdiff(`*`f`*`, `*`x`*`)`	Differentiation operator. Differentiation of *`f`* with respect to *`x`*. No chain rule for dependent variables.
`test(`*`expr`*`)`	Test function operator
`nojac(`*`expr`*`)`	No contribution to the Jacobian
`up(`*`expr`*`)`	Evaluate expression as defined in adjacent up side
`down(`*`expr`*`)`	Evaluate expression as defined in adjacent down side
`mean(`*`expr`*`)`	Mean value of expression as evaluated on adjacent boundaries
`depends(`*`expr`*`)`	True if expression depends on the solution
`islinear(`*`expr`*`)`	True if expression is a linear function of the solution
`dest(`*`expr`*`)`	Evaluate parts of an integration coupling expression on destination side.
`if(cond,expr1,expr2)`	Conditional expression evaluating the second or third argument depending on the value of the condition
`quad(f,x,a,b,tol)`	Adaptive numerical quadrature of f with respect to x