ADMMacros

Original code by Gabrielle Allen,
later enhancements by Denis Pollney

Date: 2004/11/21 12:55:34

Abstract

Provides macros for common relativity calculations, using the **ADMBase** variables.

1 Purpose

This thorn provides various macros which can be used to calculate quantities, such as the Christoffel Symbol or Riemann Tensor components, using the basic variables of thorn **ADMBase** (and **StaticConformal** if required). The macros can be used in both Fortran and C code. The macros work pointwise to calculate quantities at the grid point (`i`, `j`, `k`); it's up to you to loop over all the grid points where you want computations done. The macros are written in such a way that within any single loop, quantities which have already been calculated are automagically reused as needed in later calculations.

1.1 Finite Differencing

By default, the macros use centered 2nd order finite differencing, with 3-point finite difference molecules. That is, when finite differencing the the grid-point indices `i` ± 1, `j` ± 1, and `k` ± 1 must also be valid, and `driver::ghost_size` must be set to at least 1.

Some of the macros also support centered 4th order finite differencing; This is selected with the parameter `spatial_order`. This may be set to either 2 or 4; it defaults to 2. If it's set to 4, then 5-point finite difference molecules are used, so the grid-point indices `i` ± 2, `j` ± 2, and `k` ± 2 must also be valid, and `driver::ghost_size` must be set to at least 2. The only save way to be certain which macros support 4th order finite differencing is to check the source code; the macros which don't support it simply hard-code 2nd order finite differencing and ignore the `spatial_order` parameter.

At present 4th order finite differencing is only supported for Fortran code. (That is, at present the C versions of the macros all ignore the `spatial_order` parameter.)

2 Using ADM Macros

Each macro described in Section 4 is implemented using three include files:

`<MACRONAME>_declare.h` sets up the declarations for the internal macro variables. All the internal (hidden) variables have names beginning with the macro name. This file should be included in the declarations section of your routine.

`<MACRONAME>_guts.h` is the actual included source code which will calculate the quantities.

`<MACRONAME>_undefine.h` resets the macros. This file **must** be `#include`d at the end of every loop using macros. Without this, a second loop using macros would assume that quantities have already been calculated (and thus get wrong results).

The macros which compute derivatives also use the following variables; you should avoid using these in your code, in either lower or upper case:

```
di2, dj2, dk2        /* only in C, not in Fortran */
dt, dx, dy, dz
idx, idy, idz
i2dx, i2dy, i2dz
i12dx, i12dy, i12dz
idxx, idxy, idxz, idyy, idyz, idzz
i12dxx, i12dyy, i12dzz
i36dxy, i36dxz, i36dyz
```

To use the macros, first find the name of the macro from the table in Section 4 and put the include files in the correct place following the instructions above. Note that all ADMMacro include files are in the directory `CactusEinstein/ADMMacros/src/macro/`, so you include the macros with lines such as

```
#include "CactusEinstein/ADMMacros/src/macro/<MACRONAME>_<TYPE>.h"
```

(Recall that Cactus uses a C-style preprocessor for Fortran as well as C/C++ code; you use the same `#include`s for all these languages.)

Each variable that the macro calculates is listed in the table of Section 4. Note that these variable names are themselves macros and are case sensitive. **Always use the macro variables on the right hand sides of equations, never redefine them yourself, since they may be used in later (hidden) calculations.**

2.1 Fortran

If you are using the macros inside a Fortran function then the `i`, `j` and `k` indices are used directly.

If you're using (either directly or indirectly) any macro which computes derivatives, you also need to `#include` two additional files:

`ADM_Spacing_declare.h`
: This must be `#include`d **before** any of the other `<MACRONAME>_declare.h` files.

`ADM_Spacing.h`
: This must be `#include`d **after** all of the other `<MACRONAME>_declare.h` and **before** any of the `<MACRONAME>_guts.h` files.

The Fortran example below should make this clear(er).

2.2 C

If you are using the macros inside a C function then you must define the grid-function subscripting index `ijk`, which can be found from `i`, `j` and `k` using the macro `CCTK_GFINDEX3D(cctkGH,i,j,k)`. Of course, since `ijk` depends on `i`, `j` and `k`, you have to assign `ijk` its value **inside** the loop-over-grid-points loops.

You must also define the grid-function strides `di`, `dj` and `dk` to give the grid-function subscripting index offsets of the grid points $(i+1,j,k)$, $(i,j+1,k)$, and $(i,j,k+1)$ (respectively) relative to (i,j,k). That is, you should define `di = 1`, `dj = cctk_lsh[0]`, and `dk = cctk_lsh[0]*cctk_lsh[1]`. Since these don't depend on `i`, `j` and `k`, they can be assigned values once outside the loop-over-grid-points loops.

The C example below should make this clear(er).

Note that you should assign all these variables their values **before** `#include`ing the `<MACRONAME>_guts.h` macro (it may do calculations which use these values).

3 Examples

3.1 Fortran

This example comes from thorn `CactusEinstein/Maximal` and uses the trK macro to calculate the trace of the extrinsic curvature.

```
c     Declarations for macros.
#include "CactusEinstein/ADMMacros/src/macro/TRK_declare.h"

c we're not taking any derivatives here, but if we were,
c we would also need the following line:
#include "CactusEinstein/ADMMacros/src/macro/ADM_Spacing_declare.h"

c we're not taking any derivatives here, but if we were,
c we would also need the following line:
#include "CactusEinstein/ADMMacros/src/macro/ADM_Spacing.h"

<CUT>

c     Add the shift term: N = B^i D_i(trK).
      if ((maxshift).and.(shift_state.eq.1)) then
         do k=1,nz
            do j=1,ny
               do i=1,nx
#include "CactusEinstein/ADMMacros/src/macro/TRK_guts.h"
                  K_temp(i,j,k) = TRK_TRK
                end do
             end do
         end do
#include "CactusEinstein/ADMMacros/src/macro/TRK_undefine.h"
```

3.2 C

This function computes the curved-space Laplacian of a scalar field ϕ, $\nabla^i \nabla_i \phi = g^{ij} \partial_{ij} \phi - g^{ij} \Gamma^k_{ij} \partial_k \phi$, assuming that the partial derivatives $\partial_{ij} \phi$ and $\partial_k \phi$ have already been computed:

```
/*
 * This function computes the curved-space Laplacian of a scalar field,
 * $\del^i \del_i \phi
 *     = g^{ij} \partial_{ij} \phi - g^{ij} \Gamma^k_{ij} \partial_k \phi$
 * at the interior grid points only; it doesn't do anything at all on the
 * boundaries.
 *
 * This function uses the following Cactus grid functions:
 *    input:   dx_phi, dy_phi, dz_phi       # 1st derivatives of phi
 *             dxx_phi, dxy_phi, dxz_phi,   # 2nd derivatives of phi
 *                      dyy_phi, dyz_phi,
 *                               dzz_phi
 *    output:  Laplacian_phi
 */
void compute_Laplacian(CCTK_ARGUMENTS)
{
DECLARE_CCTK_ARGUMENTS
int i,j,k;

/* contracted Christoffel symbols $\Gamma^k = g^{ij} \Gamma^k_{ij}$ */
CCTK_REAL Gamma_u_x, Gamma_u_y, Gamma_u_z;

/* grid-function strides for ADMMacros */
const int di = 1;
const int dj = cctk_lsh[0];
const int dk = cctk_lsh[0]*cctk_lsh[1];

/* declare the ADMMacros variables for $g^{ij}$ and $\Gamma^k_{ij}$ */
#include "CactusEinstein/ADMMacros/src/macro/UPPERMET_declare.h"
#include "CactusEinstein/ADMMacros/src/macro/CHR2_declare.h"

    for (k = 1 ; k < cctk_lsh[2]-1 ; ++k)
    {
    for (j = 1 ; j < cctk_lsh[1]-1 ; ++j)
    {
    for (i = 1 ; i < cctk_lsh[0]-1 ; ++i)
    {
    const int ijk = CCTK_GFINDEX3D(cctkGH,i,j,k);   /* grid-function subscripting index for ADMMacros */
                                                     /* (must be assigned inside the i,j,k loops) */

    /* compute the ADMMacros $g^{ij}$ and $\Gamma^k_{ij}$ variables at the (i,j,k) grid point */
    #include "CactusEinstein/ADMMacros/src/macro/UPPERMET_guts.h"
    #include "CactusEinstein/ADMMacros/src/macro/CHR2_guts.h"

    /* compute the contracted Christoffel symbols $\Gamma^k = g^{ij} \Gamma^k_{ij}$ */
    Gamma_u_x =
      UPPERMET_UXX*CHR2_XXX + 2.0*UPPERMET_UXY*CHR2_XXY + 2.0*UPPERMET_UXZ*CHR2_XXZ
                            +     UPPERMET_UYY*CHR2_XYY + 2.0*UPPERMET_UYZ*CHR2_XYZ
                                                        +     UPPERMET_UZZ*CHR2_XZZ;
    Gamma_u_y =
      UPPERMET_UXX*CHR2_YXX + 2.0*UPPERMET_UXY*CHR2_YXY + 2.0*UPPERMET_UXZ*CHR2_YXZ
                            +     UPPERMET_UYY*CHR2_YYY + 2.0*UPPERMET_UYZ*CHR2_YYZ
                                                        +     UPPERMET_UZZ*CHR2_YZZ;
    Gamma_u_z =
      UPPERMET_UXX*CHR2_ZXX + 2.0*UPPERMET_UXY*CHR2_ZXY + 2.0*UPPERMET_UXZ*CHR2_ZXZ
                            +     UPPERMET_UYY*CHR2_ZYY + 2.0*UPPERMET_UYZ*CHR2_ZYZ
                                                        +     UPPERMET_UZZ*CHR2_ZZZ;

    /* compute the Laplacian */
    Laplacian_phi[ijk] =
      UPPERMET_UXX*dxx_phi[ijk] + 2.0*UPPERMET_UXY*dxy_phi[ijk] + 2.0*UPPERMET_UXZ*dxz_phi[ijk]
                                +     UPPERMET_UYY*dyy_phi[ijk] + 2.0*UPPERMET_UYZ*dyz_phi[ijk]
                                                                +     UPPERMET_UZZ*dzz_phi[ijk]
      -  Gamma_u_x*dx_phi[ijk]  -        Gamma_u_y*dy_phi[ijk]  -        Gamma_u_z*dz_phi[ijk];
    }
```

```
    }
    }

#include "CactusEinstein/ADMMacros/src/macro/UPPERMET_undefine.h"
#include "CactusEinstein/ADMMacros/src/macro/CHR2_undefine.h"
}
```

4 Macros

Macros exist for the following quantities

Calculates	**Macro Name**	**Sets variables**
All first spatial derivatives of lapse, $\alpha_{,i}$:	DA	DA_DXDA, DA_DYDA, DA_DZDA
All second spatial derivatives of lapse, $\alpha_{,ij}$:	DDA	DDA_DXXDA, DDA_DXYDA, DDA_DXZDA, DDA_DYYDA, DDA_DYZDA, DDA_DZZDA
All second covariant spatial derivatives of lapse, $\alpha_{;ij}$:	CDCDA	
All first spatial derivatives of shift, $\beta^i{}_j$:	DB	
All first covariant derivatives of the extrinsic curvature, $K_{ij;kl}$	CDK	
First covariant derivatives of the extrinsic curvature, $K_{ij;x}$, $K_{ij;y}$, $K_{ij;z}$	CDXCDK, CDYCDK, CDZCDK	
Determinant of 3-metric:	DETG	
Upper 3-metric, gij:	UPPERMET	
Trace of extrinsic curvature trK:	TRK	
Trace of stress energy tensor:	TRT	
Hamiltonian constraint:	HAMADM	
Partial derivatives of extrinsic curvature, $K_{ij,x}$, $K_{ij,y}$, $K_{ij,z}$:	DXDK, DYDK, DZDK	
First partial derivatives of 3-metric, $g_{ij,x}$, $g_{ij,y}$, $g_{ij,z}$:	DXDG, DYDG, DZDG	
All first partial derivatives of 3-metric, $g_{ij,k}$:	DG	
First covariant derivatives of 3-metric, $g_{ij;x}$, $g_{ij;y}$, $g_{ij;z}$:	DXDCG, DYDCG, DZDCG	
Second partial derivatives of 3-metric, $g_{ij,xx}$, $g_{ij,xy}$, $g_{ij,xz}$:	DXXDG, DXYDG, DXZDG, DYYDG, DYZDG, DZZDG	
All second partial derivative of 3-metric, $g_{ij,lm}$	DDG	
Ricci tensor R_{ij}:	RICCI	
Trace of Ricci tensor R:	TRRICCI	
Christoffel symbols of first kind: Γ_{cab}	CHR1	
Christoffel symbols of second kind $\Gamma^c{}_{ab}$:	CHR2	
Momentum constraints	MOMX, MOMY, MOMZ	
Source term in evolution equation for conformal metric, $\tilde{g}_{ij,t}$:	DCGDT	

5 Definitions

$$\Gamma_{cab} = \frac{1}{2}\left(g_{ac,b} + g_{bc,a} - g_{ab,c}\right) \tag{1}$$

$$\Gamma^c{}_{ab} = g^{cd}\Gamma_{dab} = \frac{1}{2}g^{cd}\left(g_{ad,b} + g_{bd,a} - g_{ab,d}\right) \tag{2}$$