REVISTA MEXICANA DE FÍSICA S **53** (4) 159–170 AGOSTO 2007

Defínción y cálculo de tensores mediante el paquete GRTensor

R.A. Sussman
Instituto de Ciencias Nucleares, Universidad Nacional Autónoma de México,
Circuito Exterior, C.U., México D.F., 04510, México,
e-mail: sussman@nucleares.unam.mx,
http://www.nuclecu.unam.mx/ sussman

Recibido el 1 de mayo de 2006; aceptado el 1 de noviembre de 2006

El paquete GRTensor permite definir, calcular y manipular tensores de rango y dimensión arbitrarios, a partir de las componentes de una métrica dada en una base de coordenadas o mediante bases de tétradas arbitrarias. Presentamos en el presente trabajo algunas aplicaciones básicas de dicho paquete a la definicion y evaluación de objetos tensoriales de uso común en Relatividad General, considerando para este objetivo ejemplos sencillos e ilustrativos.

Descriptores: Álgebra computacional; relatividad general; tensores.

The package GRTensor allows one to define, calculate and manipulate tensors of arbitrary range and dimension from the components of a metric, either in a coordinate basis or tetrad. By considering simple and instructive examples we implement basic applications of this package to define and compute tensorial objects of common usage in General Relativity.

Keywords: Computer álgebra; general relativity; tensors.

PACS: 95.30.Lz; 95.30.Sf; 98.80-k

1. Introducción

Los objetos tensoriales de rango diverso son de grán importancia en varias disciplinas de la Física Teórica y Matemáticas Aplicadas: Relatividad General, Teoría del Medio Continuo, Termodinámica e Hidrodinámica de Fenómenos de Transporte, Geometría Diferencial. Debido a la alta complejidad de los algoritmos utilizados, es sumamante útil poder aplicar programas conocidos de cómputo simbólico al cálculo y manipulación de dichos objetos. El presente trabajo ilustra cómo se efectúa esta manipulación tensorial en el paquete GRTensor[i].

El paquete GRTensor [1] es un módulo completo de subrutinas del lenguaje de programación interpretado Maple [2], cuyo propósito es definir, calcular y manipular objetos tensoriales de rango y dimensión arbitrarios. Dichas subrutinas corren en el ambiente simbólico–numérico y gráfico de Maple. Al correr GRTensor en una sesión de Maple podemos utilizar las instrucciones y comandos pre-existentes en Maple, sin embargo GRTensor cuenta con instrucciones propias (diferentes de las de Maple estándar) que llevan a definir objetos de GRTensor que solo pueden ser operados o modificados por instrucciones de GRTensor.

Es necesario mencionar que, además de las versiones programadas de GRTensor para Maple, existen versiones programadas de GRTensor que corren sobre el paquete Mathematica, en forma muy semejante a como corre sobre Maple. GRTensor puede ser utilizado en cualquiera de las plataformas de hardware compatibles con Maple o Mathematica: DOS-Windows, Apple–Macintosh, Intel–Linux, así como en las principales estaciones de trabajo con sistemas operativos Unix. Mientras que Maple y Mathematica son paquetes de software comerciales, GRTensor puede ser descargado de la red gratuitamente [1].

Al descargar GRTensor, el usuario recibe los archivos binarios que permiten activar las subrutinas del mismo (ya sea en Maple o en Mathematica[ii]. Junto con estos archivos viene la documentación necesaria para el uso de GRTensor: el manual (en archivos postscript), un sistema de ayuda en linea en hipertexto, asi como archivos de texto que contienen las componentes del tensor métrico de un vasto catálogo de espacio–tiempos conocidos (los cuales GRTensor deberá cargar para definir y calcular tensores).

En lo que resta del presente artículo describimos el funcionamiento de GRTensor, empezando por la instalción del mismo, siguiendo por la lectura de una métrica dada, así como la definición, cálculo y manipulación de varios tensores relevantes.

2. Instalación y lectura de métricas

Una vez que recibimos de la red la distribución de GRTensor, el primer paso es el proceso de su instalación como módulo asociado a Maple. Obviamente, GRTensor requiere que Maple esté funcionando en la computadora huesped. La documentación que acompaña a GRTensor contiene instrucciones precisas para la instalación del mismo en los sistemas DOS–Windows, Intel–Linux y estaciones de trabajo basadas en Unix. Para su instalación en la plataforma Apple–Macintosh (sistemas 9 y anteriores, asi como en MacOSX), sugerimos que el lector consulte la Ref. 4.

Para definir y calcular tensores arbitrarios GRTensor requiere la especificación de un tensor métrico, ya sea dado en una base de coordenadas o en una base de vectores. Además de los archivos binarios (extensión .m), la distrubución de GRTensor que recibimos de la red contiene un archivo de inicialización (MapleInit) y un directorio llamado “metrics” que contiene archivos de texto (con extensión .mpl) que especifi-

can los componentes de las métricas que deberá leer GRTensor para definir y calcular tensores.

Aunque los detalles del proceso de instalación dependen del sistema operativo de la computadora huesped, independientemente del mismo, este proceso consiste en los siguientes tres pasos:

1. Colocar el directorio “lib” que contiene los archivos binarios de GRTensor en los subdirectorios apropiados.
2. Editar el archivo de inicialización de GRTensor, dando la trayectorial local del subdirectorio “lib”, de manera que los binarios puedan ser leidos al abrir una sesión de Maple.
3. Editar el archivo de inicialización de modo que quede especificada la trayectoria local del directorio “metrics”.

Estos pasos vienen descritos en un archivo “readme”, así como en el sitio oficial de GRTensor [1]. El usuario puede también consultar [4] o solicitar ayuda al autor por correo electrónico.

Una vez terminado el proceso de instalación, al teclear la instrucción de inicialización de GRTensor

```
> grtw();
```

debe aparecer un “mensaje de bienvenida” que indica que GRTensor está listo para ser utilizado.

El siguiente paso es cargar una métrica, es decir, alguno de los archivos de texto .mpl que estan ubicados en el subdirectorio “metrics”. Es posible también introducir los componentes de la métrica (en base de coordenadas o de vectores) en forma interactiva (ver Ref. 4), sin embargo en este artículo veremos únicamente el proceso no–interactivo.

Supongamos que deseamos que GRTensor defina y calcule tensores asociados a la métrica Robertson–Walker dada en una base de coordenadas esféricas (r, θ, ϕ, t):

$$ds^2 = -dt^2 + \frac{A^2(t)[dr^2 + r^2(d\theta^2 + \sin^2\theta\, d\phi^2)]}{\left[1 + \frac{k_0}{4} r^2\right]^2}, \quad (1)$$

donde $k_0 = 0, \pm 1$. El archivo de texto con extensión .mpl que debemos crear (si es que aún no existe en el directorio “metrics”) es:

```
 Ndim := 4 :
_x1 := r :
_x2 := theta :
_x3 := phi :
_x4 := t :
g11_ := A(t)^2/(1+k0/4*r^2)^2 :
g22_ := A(t)^2*r^2/(1+k0/4*r^2)^2 :
g33_ := A(t)^2*r^2*sin(theta)^2/(1+k0/4*r^2)^2
:
g44_ := -1 :
# Metrica Robertson-Walker
# Coordenadas esfericas r,theta,phi,t
# k0 = 0,1,-1
```

Si a este archivo lo nombramos “RW.mpl”, entonces la instruccion de GRTensor que carga esta métrica es

```
> qload(RW);
```

Nótese que la extension .mpl ha sido omitida. El archivo .mpl indica el número de dimensiones, los nombres de cada coordenada, así como comentarios pertinentes que Maple no lee (lineas que inician con el caracter #). Una vez ejecutada la instrucción `qload` GRTensor responde con la métrica dada exactamente por (1). GRTensor está ahora listo para definir y calcular tensores para dicha métrica. La Fig. 1 ilustra la acción de las instrucciones `grtw` y `qload` para la métrica `RW.mpl`.

3. Tensores predefinidos: modo de cálculo

GRTensor tiene una lista amplia de tensores y objetos tensoriales de uso común predefinidos, por ejemplo, la métrica inversa y el determinante de la métrica, g^{ab}, g, los símbolos de Christoffel, $\Gamma^a{}_{bc}$, los tensores de Riemann, Ricci, Einstein, Weyl, (R^a_{bcd}, R^a_b, G^a_b, C^a_{bcd}) y todos sus objetos duales obtenidos por contracciones con la métrica (subir y bajar índices), así como los escalares que resultan de sus contracciones entre ellos: escalar de Ricci, Kretschmann, ($R = R^a_a$, $K = R^{abcd}R_{abcd}$). La lista de objetos predefinidos puede ser consultada accediendo a la hoja de ayuda en linea “grt-objects”.

Los objetos predefinidos pueden ser calculados directamente mediante la instrucción “`grcalc`”. El uso de esta instrucción determina el “modo de cálculo” de GRTensor en el que la posición de cada índice se especifica con “dn” (índice abajo) y “up” (índice arriba). Por ejemplo, si queremos que GRTensor calcule el tensor $R^a{}_b$ (tensor “mixto” de Ricci, debemos teclear

```
> grcalc(R(up,dn));
```

Para que GRtensor despliegue el tensor tecleamos

```
> grdisplay(R(up,dn));
```

De la misma forma podemos calcular y desplegar cualquier otro objeto predefinido (ver Fig. 2). La instrucción `grcalcd` calcula y despliega el tensor.

```
> grtw();
Scalar invariant library.
Last modified 25 March 1997.
`Differential Invariants`
`Last modified Jan. 20, 1995`

`Basis/tetrad related object definitions`
`Last modified 23 January 2001`
`Last built 27 May, 1999`

`Last built 27 May, 1999`
```

GRTensorII Version 1.79 (R4)

6 February 2001

Developed by Peter Musgrave, Denis Pollney and Kayll Lake

Copyright 1994-2001 by the authors.

Latest version available from: http://grtensor.phy.queensu.ca/

/grtiiOSX/metrics/

```
> qload(RW);
```

Default spacetime = RW

For the RW spacetime:

Coordinates

x(up)

$$x^{a} = [r, \theta, \phi, t]$$

Line element

$$ds^2 = \frac{A(t)^2\, dr^2}{\left(1 + \frac{1}{4} k0\, r^2\right)^2} + \frac{A(t)^2 r^2\, d\theta^2}{\left(1 + \frac{1}{4} k0\, r^2\right)^2} + \frac{A(t)^2 r^2 \sin(\theta)^2\, d\phi^2}{\left(1 + \frac{1}{4} k0\, r^2\right)^2} - dt^2$$

FIGURA 1. Input y output de las instrucciones `grtw` y `qload` para la métrica Robertson–Walker.

```
> grcalc(R(up,dn));
Created definition for R(up,dn)
```

CPU Time = 0.230

```
> grdisplay(_);
```

For the RW spacetime:

R(up,dn)

R(up, dn)

$$R^{a}{}_{b} = \left[\left[\frac{2\,k0 + 2\left(\frac{d}{dt}A(t)\right)^2 + A(t)\left(\frac{d^2}{dt^2}A(t)\right)}{A(t)^2}, 0, 0, 0\right], \left[0, \frac{2\,k0 + 2\left(\frac{d}{dt}A(t)\right)^2 + A(t)\left(\frac{d^2}{dt^2}A(t)\right)}{A(t)^2}, 0, 0\right], \left[0, 0, \frac{2\,k0 + 2\left(\frac{d}{dt}A(t)\right)^2 + A(t)\left(\frac{d^2}{dt^2}A(t)\right)}{A(t)^2}, 0\right], \left[0, 0, 0, \frac{3\left(\frac{d^2}{dt^2}A(t)\right)}{A(t)}\right]\right]$$

FIGURA 2. Input y output de las instrucciones `grcalc` y `grdisplay` para el tensor de Ricci.

En el caso de escalares (no hay índices) `grcalc` simplemente actúa sobre el nombre asignado al objeto (ver hoja de ayuda en linea "grt_objects"). Por ejemplo, el nombre asignado al escalar de Ricci es `Ricciscalar`, por lo que para calcular y visualizar este objeto tecleamos

```
> grcalc(Ricciscalar);
> grdisplay(_);
```

en donde hemos usado el caracter _ con el cual GRTensor denota que `grdisplay` es aplicada al último objeto tensorial calculado, o sea, en este caso `Ricciscalar`.

3.1. Simplificación, factorización y sustitución en Maple estándar.

Una vez calculado un tensor dado, podemos aplicar globalmente (a todas sus componentes) cualquier instrucción o rutina de manipulación algebráica de Maple. Por ejemplo, que factorice o expanda cada componente o que sustituya alguna variable o una función arbitraria que aparezca en la métrica. Para entender como GRTensor opera globalmente estas rutinas es conveniente examinar cómo estas operan en Maple estándar. Las instrucciones mas comunes de este tipo son:

- `normal`, transforma

  $\frac{a}{b}+\frac{c}{d}$ en $\frac{ad+bc}{bd}$

- `expand`, transforma

  $\frac{a+b}{c}$ en $\frac{a}{c}+\frac{b}{c}$
  $(a+b)c$ en $ac+ab$
  $(a+b)^2$ en $a^2+2ab+b^2$

- `factor`, transforma

  $a^2+2ab+b^2$ en $(a+b)^2$

Además de estas instrucciones está "`simplify`", la cual es equivalente a normal o a factor (a menudo estas instrucciones dan el mismo resultado).

Para manipulaciones mas especializadas Maple usa la instrucción `simplify` con una opción adecuada, por ejemplo, si X es una expresión con funciones trigonométricas, es conveniente simplificarla con la opción "trig":

```
> simplify(X, trig);
```

O bien, para que Maple transforme $\sqrt{a^2}$ en a (asumir que a es real positivo), o que exprese $(\mathrm{e}^a)^2$ como e^{2a}, usamos la instrucción simplify con la opción "symbolic". Para mayor detalle sobre estas simplificaciones ver Ref. 4.

Una manipulación algebráica muy utilizada es la sustitución de una variable por otra (o por un valor numérico). En el caso de la métrica RW, tenemos una función arbitraria $A(t)$, por lo que podría ser útil el poder sustituir esta función por un ansatz o una función explícita (por ejemplo $A(t)=t^{1/2}$). La instrucción de sustitución en Maple estándar es `subs` y su sintaxis es muy simple: para sutituir $x=k\pi/3$ en la expresión $x\,\sin x$, tecleamos:

```
> subs(x = k*Pi/3, x*sin(x) );
```

Si sustituimos una forma explícita de $A(t)$ en una expresión que contenga a esta función y derivadas de la misma, entonces para que se evaluen las derivadas debemos aplicar la instrucción `eval` después de haber sustituido con `subs`. Por ejemplo, si queremos sustituir $A(t)=t^{2/3}$ en la expresión $1+\dot{A}/A^2$ (con $\dot{}=d/dt$), le asignamos el nombre F a esta expresión y la sustitución y evaluación se da mediante:

```
> F := 1+diff(A(t),t)/A(t)^2;
> subs( A(t) = t^(2/3), F);
> eval(%);
```

donde el símbolo % indica que `eval` actúa sobre el resultado de la instrucción inmediatamente anterior.

3.2. Simplificación, factorización y sustitución con GRTensor.

Sin embargo, las instrucciones de Maple estándar que hemos descrito anteriormente no pueden ser aplicadas directamente a objetos calculados por GRTensor. Por lo tanto, una opción es transformar objetos de GRTensor en objetos de Maple (arreglos), lo cual discutiremos mas adelante. La otra opción es aplicar globalmente las instrucciones de Maple con instrucciones propias de GRTensor. Por ejemplo, si queremos aplicar "`expand`" a todas las componentes de `R(up,dn)` (o sea, $R^a{}_b$), podemos usar la instrucción `gralter` que puede trabajar en forma interactiva

```
> gralter(R(up,dn));
```

apareciendo inmediatamente un menú interactivo de opciones que identifica cada instrucción predefinida con un número. El número 6 identifica a `expand`, por lo que al teclear este número, esta instrucción es aplicada a cada componente de `R(up,dn)`. Para usar `gralter` en forma no–interactiva (cuando ya se conoce las instrucciones que aplica) simplemente se teclea el nombre de la instrucción (o su número) como segundo argumento de `gralter`.

Una opción muy útil de `gralter` es `autoAlias`, cuyo efecto es reemplazar $A(t)$ simplemente por A, mientras que sus derivadas $dA(t)/dt$ y $d^2A(t)/dt^2$ son respectivamente reemplazadas por A_t y A_{tt}. Este efecto permite tratar con expresiones mas compactas por lo que es de gran ayuda para la visualización de componentes tensoriales. Como ejemplo, aplicamos `gralter` con la opción `autoAlias` seguido inmediatamente de `expand` (ver Fig. 3):

```
> gralter(R(up,dn), autoAlias);
```

```
> gralter(_, expand);
> grdisplay(_);
```

Una variante interesante es la instrucción `grcalcalter`, que calcula un objeto predefinido y aplica (conforme calcula) las instrucciones contenidas en `gralter`.

Otra instrucción para aplicar cualquier rutina, función o instrucción estandar de Maple en forma global a objetos de GRTensor es `grmap`. Esta instrucción de es sumamente poderosa y versatil, sin embargo su sintaxis es más complicada que `gralter`. Como ejemplo (ilustrado por la Fig. 4), veamos la sustitución global de $A(t) = t^{1/3}$ y $k_0 = 0$ en el tensor $R^a{}_b$ (ya modificado por `gralter`, ver Fig. 3) tecleamos

```
> grmap(R(up,dn),subs,A(t)=t^(1/3),k0=0,
'x');
> grdisplay(_);
```

Para consultar otros ejemplos de aplicación de `grmap` referimos al lector a la hoja de ayuda de dicha instrucción, así como al manual incluido en la distribución de GRTensor (disponible en el sitio de GRTensor [1]) o bien a la Ref. 4.

3.3. Extracción de componentes individuales.

Una vez calculado, si deseamos extraer una compenente específica, por ejemplo la componente R^r_r del tensor R^a_b (componente "1,1" de `R(up,dn)`), tecleamos

```
> grcomponent(R(up,dn),[1,1]);
```

Una vez ejecutada esta instrucción, la componente extraída es una expresión de Maple estándar, por lo que es posible aplicarle directamente cualquier instrucción estandar de Maple, por ejemplo, asignarle un nombre (por ejemplo "R11"):

```
> grcomponent( R(up,dn),[1,1]);
> R11:=%;
```

o bien, aplicar cualquiera de las instrucciones de simplificación mencionadas anteriormente. Obviamente, para extraer un objeto vectorial (tensor de un índice) la componente se especifica con un solo número dentro del corchete [ ]. Para escalares no es necesario dar el número de una componente:

```
> grcomponent(Ricciscalar);
```

con lo cual es posible asignarle un nombre o aplicar cualquier instrucción de Maple estándar.

3.4. Pasar objetos de GRTensor a objetos de Maple

En vez de extraer componentes uno a uno para poderlos tratar como objetos de Maple estándar, podemos transformar globalmente objetos de GRTensor en objetos de Maple llamados "arreglos", los cuales se pueden manipular como objetos matriciales. La instrucción de GRTensor que lleva a cabo esta transformación es `grarray`. Por ejemplo, si queremos construir un arreglo de Maple con nombre asignado "Rmix" que contenga las componentes ya calculadas de `R(up,dn)` tecleamos

```
> Rmix :=  grarray( R(up,dn));
```

La componente R^r_r (o 1,1) se obtiene inmediatamente tecleando:

```
> Rmix[1,1];
```

Una vez que tenemos un objeto de GRTensor definido como arreglo de Maple estándar, lo podemos manipular o simplificar usando las instrucciones usuales de Maple para arreglos. Si aplicamos `grarray` a un objeto vectorial, obtenemos un vector de Maple estándar y entonces las componentes se extraen especificando un solo número, mientras que `grarray` aplicado a un escalar simplemente da una sola expresión (arreglo de orden cero).

4. Modo de definición de tensores

Si necesitamos trabajar con un objeto tensorial que no está en la lista de tensores predefinidos, necesitamos entonces definirlo. La sintaxis del modo de definición de GRTensor es diferente a la del modo de cálculo que vimos en la sección anterior. Obviamente, necesitamos definir un tensor antes de poder calcularlo en el modo de cálculo.

Usaremos como ejemplo la definición de una 4-velocidad u^a y la construcción subsecuente del tensor de momento–energía del fluido perfecto:

$$T^{ab} = [\mu(t) + p(t)]\, u^a u^b + p(t)\, g^{ab}, \tag{2}$$

ambos tensores compatibles con la métrica Robertson–Walker (1). Primero definimos la 4-velocidad comóvil dada ya sea por

$$u^a = \delta^a_t, \tag{3}$$

o equivalentemente como

$$u^a = [0, 0, 0, 1], \tag{4}$$

(recordemos que t es la cuarta coordenada). GRTensor permite definir u^a de ambas formas através de la instrucción `grdef`:

```
> grdef('u{^a} := kdelta{^a $t }');
```

```
> gralter(_, autoAlias);
Component simplification of a GRTensorII object:

Applying routine autoAlias to object R(up,dn)
Warning, alias or macro A[t] defined in terms of A
                                                    CPU Time = 0.
> gralter(_, expand);
Component simplification of a GRTensorII object:

Applying routine expand to object R(up,dn)
                                                    CPU Time = 0.010
> grdisplay(_);
```

For the RW spacetime:

R(up,dn)

R(up, dn)

$$R^{a}_{\ b} = \begin{bmatrix} \frac{2\,k0}{A^2} + \frac{2\,A_t^{\,2}}{A^2} + \frac{A_{t,t}}{A} & 0 & 0 & 0 \\ 0 & \frac{2\,k0}{A^2} + \frac{2\,A_t^{\,2}}{A^2} + \frac{A_{t,t}}{A} & 0 & 0 \\ 0 & 0 & \frac{2\,k0}{A^2} + \frac{2\,A_t^{\,2}}{A^2} + \frac{A_{t,t}}{A} & 0 \\ 0 & 0 & 0 & \frac{3\,A_{t,t}}{A} \end{bmatrix}$$

FIGURA 3. Modificación del tensor de Ricci mediante `gralter` con las opciones `autoAlias` y `expand`.

```
> grmap(_, subs, A=t^(1/3), k0=0, 'x');
Applying routine subs to R(up,dn)
> grdisplay(_);
```

For the RW spacetime:

R(up,dn)

R(up, dn)

$$R^{a}_{\ b} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{2}{3\,t^2} \end{bmatrix}$$

FIGURA 4. Modificación del tensor de Ricci mediante `gralter` con las opciones `autoAlias` y `expand`.

o bien

```
> grdef('u{^a} := [0,0,0,1]');
```

las cuales son muy parecidas a como de escribe u^a en (3) y (4). Del lado izquierdo del símbolo de asignación (`:=`) escribimos el nombre que será asignado al tensor que definiremos. En este caso usamos `u`, pues ninguno de los tensores predefinidos usa este símbolo (en caso contrario sería necesario escoger otro símbolo para evitar un conflicto de nombres)[iii]. Cada índice "contravariante" (superíndice) se escribe precedido con el símbolo `^`, mientras que los índices "covariantes" (subíndice) se escriben como texto normal, todos ellos dentro de llaves { } que siguen al nombre del tensor. Del lado derecho de `:=` escribimos la fórmula o regla de correspondencia del tensor [como en (3)] o como una lista de componentes [como en (4)], lo cual es solo aplicable al caso de tensores de rango 1 (vectores o 1–formas como u^a o u_a). Nótese que el tensor "delta de Kronecker" puede definirse con un índice dado por una coordenada (en este caso t).

Una vez definida u^a, de una forma u otra, necesitamos calcularla. En modo de cálculo, el tensor definido como `u{^a}` pasa a ser `u(up)`, por lo que se calcula mediante

```
> grcalc(u(up));
```

Ahora, ya que hemos definido y calculado u^a y dado que g^{ab} ya está predefinido, podemos construir y definir el tensor T^{ab} dado por (2) de la siguiente forma:

```
> grdef('T{^a ^b} :=
(mu(t)+p(t))*u{^a}*u{^b}+p(t)*g{^a ^b}');
```

la cual es (como para u^a) muy parecida a como se define T^{ab} en notación matemática en (2). Como no hay conflicto de nombres, usamos el nombre `T` y del lado derecho de `:=` escribimos la regla de correspondencia dada por (2). Remarcamos que en la definicion de este tensor `grdef` denota con el mismo símbolo * a productos tensoriales (como $u^a u^b$) y a productos por un escalar (como $p(t)\, g^{ab}$).

Habiendo definido `T{^a ^b}` mediante `grdef` tenemos un nuevo tensor que, en modo de cálculo, se denota `T(up,up)`. Para calcularlo usamos `grcalc` aplicado a `T(up,up)`. Una vez definido el tensor T^{ab}, podemos calcular y visualizar directamente con `grcal` y `grdisplay` (sin tener que definirlos previamente) los tensores duales asociados $T^a{}_b$ o T_{ab}:

```
> grcalc(T(up,dn));
> grdisplay(_);
```

Así como formamos T^{ab} a partir de u^a y g^{ab}, podemos definir y calcular las ecuaciones de campo de Einstein $G^a{}_b = \kappa T^a{}_b$, pues el tensor de Einstein $G^a{}_b = R^a{}_b - (1/2)\, R\, \delta^a{}_b$ está predefinido y ya tenemos T^{ab} definido y calculado. Las ecuaciones de Einstein se definen y calculan mediante:

```
>  grdef('EE{^a b} := G{^a b}= kappa*T{^a
b}');
> grcalc(EE(up,dn));
> gralter(_, expand);
> gralter(_, autoAlias);
> grdisplay(_);
```

donde hemos modificado el nuevo tensor mediante `gralter` con las opciones `expand` y `autoAlias`. Nótese cómo GRTensor permite definir ecuaciones tensoriales, además de tensores individuales. Es importante remarcar que, tanto en Maple estándar como en GRTensor, el símbolo `:=` no denota una igualdad (ecuación), sino una asignación de un nombre a una variable o expresión, mientras que el símbolo `=` denota simplemente una igualdad. Una vez definidas las ecuaciones de Einstein como una ecuación tensorial, podemos aplicar rutinas de simplificación, sustitución o manipulación, tal y como se aplican a tensores predefinidos (ver sección anterior):

```
> gralter(EE(up,dn), expand);
> grdisplay(_);
```

GRTensor permite también definir tensores en cuya regla de correspondencia aparecen derivadas covariantes. Por ejemplo, el 4–vector formado por la divergencia $J^a \equiv T^{ab}{}_{;b}$, queda definido y calculado mediante:

```
> grdef('J{^a} := T{^a ^b;b}');
> grcalc(J(up));
> grdisplay(_);
```

Nótese que aunque la derivada covariante aparece como subíndice en $T^{ab}{}_{;b}$, en `grdef` el índice de ésta se coloca junto al superíndice `^b`. Si en vez de la divergencia $J^a \equiv T^{ab}{}_{;b}$ quisieramos definir y calcular la derivada covariante $J^{ab}{}_{c} \equiv T^{ab}{}_{;c}$, la regla de correspondencia en `grdef` sería: `J{^a ^b;c }:=T{^a ^b;c}`. GRTensor también permite definir y calcular objetos tensoriales obtenidos mediante derivadas ordinarias, escribiendo en ese caso una coma "`,`" en vez del punto y coma "`;`". Estos objetos tensoriales se calculan y manipulan mediante las instrucciones `grcalc, gralter, grmap, grdisplay`, tal y como se hace con los tensores predefinidos.

Un ejemplo muy útil e interesante es el tensor de momento–energía de un campo escalar Φ con potencial $V = V(\Phi)$:

$$T_{ab} = \Phi_{,a}\, \Phi_{,b} - \left[\frac{1}{2}\Phi_{,c}\Phi^{,c} + V(\Phi)\right] g_{ab}. \qquad (5)$$

Para la métrica RW necesariamente $\Phi = \Phi(t)$, por lo que $V = V(t)$. Tenemos entonces $\Phi_{,a} = [0, 0, 0, \Phi_{,t}]$. Construimos primero este objeto tensorial de rango 1:

```
> grdef('DPhi{a} := [0,0,0,diff(Phi(t),t)]');
> grcalc(DPhi(dn));
> grdisplay(_);
```

Inmediatamente podemos definir (5), pues el tensor métrico está predefinido:

```
> grdef('TS{a b} :=
DPhi{a}*DPhi{b}-(1/2*DPhi{c}*DPhi{^c}+V(t))*g{a
b}');
> grcalc(TS(up,dn));
> grdisplay(_);
```

La ecuación de Klein–Gordon sigue de calcular la divergencia $J^a = T^{ab}{}_{;b}$ con el tensor (5):

```
> grdef('JS{^a} := TS{^a ^b;b}');
> grcalc(JS(up));
> gralter(JS(up), expand);
> gralter(JS(up), autoAlias);
> grdisplay(_);
```

Si tomamos en cuenta que $\dot{V}/\dot{\Phi} = dV/d\Phi$, obtenemos la forma usual de la ecuación de Klein–Gordon para la geometría Robertson–Walker.

5. Definición y manipulación de tensores mas complicados.

Para ilustrar ejemplos mas complicados usaremos una métrica menos restringida que la de Robertson–Walker (1). Consideremos la métrica general con simetría esférica dada por el elemento de linea

$$ds^2 = -A^2\,dt^2 + B^2\,dr^2 + Y^2\,(d\theta^2 + \sin^2\theta\,d\phi^2), \quad (6)$$

donde A, B y Y son funciones arbitrarias de (t, r). El archivo de texto equivalente al archivo "`RW.mpl`" lo llamamos "`SS.mpl`" y viene dado por:

```
 Ndim := 4 :
_x1 := r :
_x2 := theta :
_x3 := phi :
_x4 := t :
g11_ := B(t,r)^2 :
g22_ := Y(t,r)^2 :
g33_ := Y(t,r)^2*sin(theta)^2 :
g44_ := -A(t,r)^2 :
# Metrica general con simetria esferica
```

Por lo tanto, en una nueva sesión de Maple repetimos el proceso de carga de GRTensor y de esta nueva métrica tal y como procedimos en las sección II con las las instrucciones `grtw` y `qload`, pero ahora la métrica (6):

```
> grtw();
> qload(SS);
```

A continuación podemos calcular para esta métrica cualquiera de los objetos tensoriales predefinidos, asi como definir y calcular tensores nuevos.

5.1. Tensores con simetrías y componentes especiales

Muy a menudo un tensor de rango mayor o igual a 2 presenta simetrías especiales entre sus índices. Veamos el caso de rango 2: a partir de cualquier tensor V_{ab} podemos construir su parte simétrica y antisimétrica mediante:

$$V_{(ab)} \equiv \frac{1}{2}\,(V_{ab} + V_{ba}), \qquad \text{parte simétrica} \quad (7)$$

$$V_{[ab]} \equiv \frac{1}{2}\,(V_{ab} - V_{ba}), \quad \text{parte anti–simétrica} \quad (8)$$

de modo que un tensor totalmente simétrico o totalmente antisimétrico satisface, respectivamente, $V_{ab} = V_{(ab)}$ o $V_{ab} = V_{[ab]}$. GRTensor permite automáticamente definir y calcular tensores con estos tipos de simetría: basta simplemente con usar los paréntesis adecuados `(...)` o `[...]` en la definición de `grdef`.

Un ejemplo clásico de un tensor totalmente antisimétrico es el tensor de Maxwell dado en términos del potencial vectorial electromagnético A_a por:

$$F_{ab} = A_{a;b} - A_{b;a} = 2A_{[a;b]} \quad (9)$$

En particular, para la métrica (6) este tensor tiene solo dos componentes diferentes de cero: $F_{tr} = -F_{rt} = \psi$, donde $\psi = 2A_{[t,r]}$. Por lo tanto, para esta métrica este tensor se puede definir como: $F_{ab} = \psi\,(\delta^r_a\,\delta^t_b - \delta^t_a\,\delta^r_b) = \psi\,\delta^r_{[a}\,\delta^t_{b]}$, donde los paréntesis rectangulares `[... ]` indican anti–simetrización. En GRTensor este tensor anti–simétrico se define y calcula mediante las instrucciones:

```
> grdef('F{a b} :=
psi(t,r)* kdelta{[a ^$t}*kdelta{b] ^$r}');
> grcalc(F(dn,dn));
> grdisplay(_);
```

Nótese cómo GRTensor reconoce el símbolo de antisimetrización de índices `[a .. b]` en `grdef`. También, estas instrucciones ilustran cómo es posible definir el tensor "delta de Kronecker" con un índice "fijo", o sea dado por una coordenada en particular (anteponiendo el símbolo $ antes del nombre t o r en `grdef`).

También es importante remarcar que GRTensor también reconoce el símbolo de simetrización de indices `(a ..b)` dentro de `grdef`, de manera que el tensor $u_{(a;b)} \equiv (1/2)(u_{a;b} + u_{b;a})$ puede ser rápidamente definido y calculado a partir de un vector 4–velocidad u^a. El primer paso es definir, calcular y normalizar u^a (ya que $u_b u^b = -1$).

Si consideramos una velocidad comóvil, tenemos:

```
> grdef('u{^a} :=[0,0,0,1]');
> grcalc(u(up));
> grnormalize(u(up,-1));
```

La instrucción `grnormalize(u(up,-1));` normaliza el vector u^a previamente calculado para que su norma sea $u_a u^a = -1$ (sin embargo puede ser aplicada a cualquier vector calculado, ya sea temporal o espacial, en cuyo caso se normaliza a 1). Para la métrica (6) `grnormalize(u(up,-1));` lleva al vector u^a cuya única componente diferente de cero es $u^t = 1/A$. El siguiente paso es definir y calcular $U_{ab} \equiv u_{(a;b)}$

```
> grdef('U{a b} :=u{(a;b)}');
> grcalc(U(up,dn));
> gralter(_, autoAlias);
> grdisplay(_);
```

en donde hemos usado la opción `autoAlias` para expresar $A(t)$ como A y sus derivadas como subíndices.

Otro ejemplo interesante es el definir un tensor de traza cero ortogonal a la 4–velocidad. Sea P_{ab} tal que $P_{ab}u^b = 0$ y $P^c{}_c = 0$. Para la métrica (6) y la 4–velocidad $u^a = \sqrt{-g^{tt}}\,\delta^a_t = (1/A)\,\delta^a_t$, este tensor podría tener como únicas componentes diferentes de cero a: $P^r{}_r = -2\Pi$, $P^\theta{}_\theta = P^\phi{}_\phi = \Pi$, donde $\Pi = \Pi(t,r)$ es una función arbitraria. Por lo tanto, tenemos la expresión tensorial

$$P^a{}_b = -2\Pi\,\delta^a_r\delta^r_b + \Pi\,\delta^a_\theta\delta^\theta_b + \Pi\,\delta^a_\phi\delta^\phi_b, \qquad (10)$$

la cual podemos definir y calcular en GRTensor mediante

```
> grdef('P{^a b} :=
-2*PI(t,r)*kdelta{^a $r}*kdelta{^$r b}
+ PI(t,r)*kdelta{^a $theta}*kdelta{^$theta b}
+ PI(t,r)*kdelta{^a $phi}*kdelta{^$phi b}');
> grcalc(P(up,dn));
> grdisplay(_);
```

El tensor de momento–energía mas general que podemos asociar a la métrica (6) es

$$T_{ab} = (\mu + p)u_a u_b + p g_{ab} + P_{ab} + 2q_{(a}u_{b)}, \qquad (11)$$

donde μ, p (densidad de energía y presión isotrópica) son funciones de (t,r), P_{ab} es el tensor definido anteriormente (presiones anisotrópicas) y q_a (flujo de energía) satisface $q_a u^a = 0$. Para la métrica (6) este covector es $q_a = Q\,\delta^r_a$, donde $Q = Q(t,r)$ es una función arbitraria. Definimos y calculamos primero q_a:

```
> grdef('q{a} :=Q(t,r)*kdelta{a ^$r}');
> grcalc(q(dn));
```

e inmediatamente definimos y calculamos T_{ab} dado por (11):

```
> grdef('T{a b} :=
(mu(t,r)+p(t,r))*u{a}*u{b} + p(t,r)*g{a
b}
+ P{a b} + 2*q{(a}*u{b)}');
> grcalc(T(up,dn));
```

Las ecuaciones de Einstein ($G^a{}_b = \kappa T^a{}_b$) forman un objeto tensorial que se define y calcula como:

```
> grdef('EE{^a b} := G{^a b} = kappa*
T{^a b}');
> grcalc(EE(up,dn));
> gralter(_, autoAlias);
> grdisplay(_);
```

La divergencia $T^{ab}{}_{;b}$ se define y calcula tal y como lo hicimos para la métrica Robertson–Walker:

```
> grdef('J{^a} := T{^a^b;b}');
> grcalc(J(up));
> gralter(_, autoAlias);
> grdisplay(_);
```

Sin embargo, este vector es ahora mucho mas complicado que el calculado para (1).

Para tensores de rango mayor que 2, la simetría o antisimetría sobre dos índices no–contiguos se denota mediante barras verticales. Por ejemplo, si el tensor V_{abc} es simétrico o antisimétrico con respecto a los índices a y c, esto se denota como

$$V_{(a|b|c)} \quad \text{ó} \quad V_{[a|b|c]}. \qquad (12)$$

Podemos formar un tensor de rango 3 através de $V_{abc}{=}P_{ab;c}$, donde P_{ab} es el tensor de presiones anisotrópicas de (11). Si queremos definir la parte simétrica de este tensor con respecto a los índices a y c, primero definimos y calculamos V_{abc}, para luego definir y calcular su parte simétrica

```
> grdef('V{a b c} := P{a b;c}');
> grcalc(V(dn,dn,dn));
> grdef('VS{(a b c} := V{(a |b| c)}');
> grcalc(VS(dn,dn,dn));
> gralter(_, autoAlias);
> grdisplay(_);
```

Nótese cómo GRTensor reconoce las barras verticales `|..|` en `grdef`.

5.2. Derivadas covariantes y operadores diferenciales

Dado un objeto tensorial previamente definido (o predefinido) y ya calculado, podemos calcular su derivada covariante sin tener que definir un nuevo tensor. Por ejemplo, si queremos calcular el tensor dado por la derivada covariante $R^a{}_{b;c}$,

no necesitamos definirlo con `grdef`, lo podemos calcular directamente agregando un tercer índice `cdn`:

```
> grcalc(R(up,dn,cdn));
```

Lo mismo podemos hacer con la derivada covariante de cualquier objeto tensorial definido y calculado con `grdef` y `grcalc`, por ejemplo, para los tensores u^a y P_{ab} ya calculados, podemos calcular directamente $u^a{}_{;b}$ o $P_{ab}{}^{;c}$, agregando un índice extra `cdn` o `cup` (según sea el caso):

```
> grcalc(u(up,cdn));
> grcalc(P(dn,dn,cup));
```

También es posible calcular objetos tensoriales formados mediante derivadas ordinarias, como por ejemplo $R^a{}_{b,c}$ o $P_{ab}{}^{,c}$. Estos objetos se calculan también directamente usando como índices extra `pdn` o `pup`, según sea el caso.

GRTensor tiene predefinidos varios operadores diferenciales que actúan sobre objetos tensoriales, como por ejemplo el operador delambertiano $\Box$ y la derivada de Lie $\pounds$ a lo largo de un vector v^a. Si consideramos un tensor A^{ab}, estos operadores son:

$$\Box A^{ab} = g^{cd} A^{ab}{}_{;c;d}, \tag{13}$$

$$\pounds_v A_{ab} = A_{ab\,;c} v^c + A_{bc} v^c{}_{;a} + A_{ab} v^c{}_{;b}. \tag{14}$$

Suponiendo que A_{ab} ya ha sido definido y calculado, las ecuaciones anteriores se calculan con GRTensor mediante

```
> grcalc(Box[A(up,up)]);
> grcalc(LieD[v(up),A(dn,dn)]);
```

En particular, la condición que determina si v^a es un vector de Killing (ya sea un vector de Killing propio o vector conforme de Killing) viene dada por la derivada de Lie con respecto a v^c, pero aplicada al caso particular en que $A_{ab} = g_{ab}$. Tomando en cuenta que $g_{ab;c} = 0$ obtenemos a partir de (14) la condición siguiente:

$$\pounds_v g_{ab} - \lambda g_{ab} = v_{(a;b)} - \lambda g_{ab} = 0, \tag{15}$$

donde λ es un escalar. El vector v^a es vector de Killing propio si $\lambda = 0$, lo cual define una isometría, mientras que si λ es una constante ($\neq 0$) v^a define una simetría homotética (auto–similaridad). GRTensor permite para un vector arbitrario v^a verificar si se cumple la condición (15), dando el valor de λ (si no es cero), mediante la instrucción:

```
> grcalc ( KillingTest[v] );
```

Otros operadores diferenciales son derivadas direccionales a lo largo de vectores base, así como `Dsq` y `CDsq` asociados respectivamente a $\Phi_{,a}\Phi^{,a}$ y $\Phi_{;a}\Phi^{;a}$ para un escalar previamente definido $\Phi = \Phi(x^a)$.

5.3. Tensores asociados a una 4-velocidad

Dado un campo de 4-velocidades u^a podemos definir varios tensores que surgen de la descomposición invariante de $u_{a;b}$:

$$\Theta \equiv u^a{}_{;a}, \quad \text{escalar de expansión}$$

$$\dot{u}_a \equiv u_{a;b} u^b, \quad \text{4–aceleración}$$

$$\sigma_{ab} \equiv u_{(a;b)} + \dot{u}_a u_b - \frac{\Theta}{3} h_{ab}, \quad \text{deformación (shear)}$$

$$\omega_{ab} \equiv u_{[a;b]} - \dot{u}_a u_b, \quad \text{vorticidad}$$

$$E_{ab} \equiv C_{abcd} u^c u^d, \quad \text{Tensor “eléctrico” de Weyl} \tag{16}$$

Estas cantidades tienen una clara interpretación física y geométrica y son muy útiles en el estudio de espacio–tiempos cosmológicos. Estos objetos tensoriales ya están predefinidos por lo que, habiendo definido y calculado u^a, los podemos calcular directamente con `grcalc`. Si usamos `u(up)` previamente calculado para (6), las siguientes instrucciones nos permiten calcular estos objetos (en el orden en que aparecen en (16)):

```
> grcalc(expscalar[u]);
> grcalc(acc[u](dn));
> grcalc(shear[u](dn,dn));
> grcalc(vor[u](dn,dn));
> grcalc(E[u](dn,dn));
```

Asociados a los tensores σ_{ab}, ω_{ab} tenemos a los escalares

$$\sigma = [\sigma_{ab}\sigma^{ab}]^{1/2}, \qquad \omega = [\omega_{ab}\omega^{ab}]^{1/2}, \tag{17}$$

los cuales se calculan con:

```
> grcalc(shear[u]);
> grcalc(vor[u]);
```

Otras cantidades asociadas a un vector u^a son la “parte magnética” del tensor de Weyl y la ecuación de Raychaudhuri:

$$u^a \Theta_{,a} + \frac{\Theta^2}{3} + \dot{u}^a{}_{;a} - \dot{u}^a \dot{u}_a + \sigma^2 - \omega^2 = \frac{\kappa}{2}(\mu + 3p). \tag{18}$$

cuyo lado izquierdo está completamente determinado en términos de los objetos (16) y (17).

Aunque hemos usado a la 4–velocidad u^a como ejemplo del cálculo de los tensores en (16), (17) y (18), podríamos haber utilizado cualquier campo vectorial v^a normalizado, ya sea temporal ($v^a v_a = -1$) o espacial ($v^a v_a = 1$), pero no nulo o tipo–luz ($v^a v_a = 0$).

6. Tensores en una base no–coordenada.

Hasta ahora hemos visto cómo GRTensor permite definir y calcular objetos tensoriales definidos en una base de coordenadas (base holonómica). También es posible definir y calcu-

lar tensores en términos de bases o tétradas de vectores o 1–formas anholonómicas. Como ejemplo, veamos el espacio–tiempo de Schwarzschild

$$ds^2 = -\left(1-\frac{2M}{r}\right)dt^2 + \left(1-\frac{2M}{r}\right)^{-1}dr^2 + r^2\left(d\theta^2+\sin^2\theta\, d\phi^2\right). \quad (19)$$

Para la base ortonormal de 1–formas:

$$\omega_a^{(1)} = \left[\left(1-\frac{2M}{r}\right)^{-1/2}, 0, 0, 0\right], \quad (20)$$

$$\omega_a^{(2)} = [0, r, 0, 0], \quad (21)$$

$$\omega_a^{(3)} = [0, 0, r\sin\theta, 0], \quad (22)$$

$$\omega_a^{(4)} = \left[0, 0, 0, \left(1-\frac{2M}{r}\right)^{1/2}\right], \quad (23)$$

la métrica (19) toma la forma ortonormal lorenziana:

$$\eta^{(a)(b)} = \omega_c^{(a)}\omega_d^{(b)}g^{cd} = \text{diag}[1,1,1,-1], \quad (24)$$

donde g^{ab} es el inverso matricial de la métrica dada en (19) y el índice de la tétrada toma los valores $(a) = 1, 2, 3, 4$.

Todos los tensores predefinidos que calculamos en bases de coordenadas se calculan para la base (23) y la métrica (24). Sin embargo, GRTensor debe cargar (23) y (24) leyendo un archivo de texto diferente al usado en las métricas dadas en base de coordenadas (aunque dicho archivo debe estar también ubicado en el mismo subdirectorio "metrics" que contiene los archivos de las métricas en base de coordenadas). Para el espacio–tiempo de Schwarzschild con la tétrada (23) este archivo debe tener la forma:

```
 Ndim := 4 :
_x1 := r :
_x2 := theta :
_x3 := phi :
_x4 := t :
eta11_ := 1 :
eta22_ := 1 :
eta33_ := 1 :
eta44_ := -1 :
bd11_ := (1-2*M/r)^(-1/2) :
bd22_ := r :
bd33_ := r*sin(theta) :
bd44_ := (1-2*M/r)^(1/2) :
# Metrica de Schwarzschild en base ortonormal
```

Si a este archivo lo llamamos `schwb.mpl`, entonces para cargar GRTensor y esta métrica tecleamos

```
> grtw();
> qload(schwb);
```

Una vez cargada la métrica (24) y la base (23), la instrucción de GRTensor para calcular (por ejemplo) el tensor de Riemann $R^{(a)}{}_{(b)(c)(d)}$ es

```
> grcalc(R(bup,bdn,bdn,bdn));
```

Como muestra el ejemplo, tenemos exactamente la misma sintaxis que en la base de coordenadas, excepto que en vez de denotar los índices como `up` y `dn` los denotamos como `bup` y `bdn`, mientras que para tensores nuevos formados con derivadas covariantes se agregan índices `cbup` o `cbdn` según sea el caso (ver hoja de ayuda grt_basis).

Para definir tensores nuevos mediante `grdef` usamos exactamente la misma sintaxis que en los casos de bases coordenadas, excepto que los índices se escriben en paréntesis: `(a)` o `^(a)`, respectivamente para índices covariantes y contravariantes. Por ejemplo, definimos el vector $W^{(a)} = [r, 0, 0, t]$ y lo calculamos junto con su derivada covariante $W^{(a)}{}_{;(b)}$

```
> grdef('W^(a):=[r,0,0,t]');
> grcalc(W(bup));
> grcalc(W(bup,cbdn));
```

Ahora definimos y calculamos el tensor

$$Z_{(a)(b)(c)} = R_{(a)(b)(c)(d)}W^{(d)}$$

y su derivada covariante $Z_{(a)(b)(c)}{}^{;(d)}$

```
> grdef('Z{(a)(b)(c)}:=R{(a)(b)(c)(d)*W^(d)}');
> grcalc(Z(bdn,bdn,bdn));
> grcalc(Z(bdn,bdn,bdn,cbup));
```

Hemos usado como ejemplo una tétrada ortonormal, pero también es posible utilizar tétradas de Newman–Penrose. Tanto para uno como para el otro (o cualquier) tipo de tétradas GRTensor tiene predefinidos escalares invariantes asociados a los tensores de Riemann y Weyl, ligados a varias clasificaciones invariantes de espacio–tiempos, incluyendo la clasificación de Petrov (ver hojas de ayuda grt_basis y gr_invars).

7. Otras instrucciones importantes.

En esta sección ofrecemos un resumen de algunas instrucciones útiles que no fueron descritas en las secciones anteriores.

- `grapply`

  Aplica una función o rutina de Maple a todo componente, pero (a diferencia de `grmap`) **no modifica la forma calculada del tensor en la memoria de GRTensor**. La salida de esta instrucción es a la hoja de trabajo o a un archivo de texto. Es particularmente útil para construir una salida de LaTeX de tensores calculados y modificados através de la función "latex", que genera el código de LaTeX de cualquier objeto evaluado en una sesión de Maple. Por ejemplo, si hemos

calculado el tensor de Ricci, R_{ab}, para alguna métrica, podemos generar el código LaTeX de dicho objeto y desplegarlo en la pantalla mediante:

```
> grapply ( R(dn,dn), latex, 'x' );
```

o bien, guardarlo en un archivo de texto con la opción "file =" especificando la trayectoria del archivo:

```
> grapply ( R(dn,dn), latex, 'x',
file='trayectoria' );
```

- `grcalc1(Tensor(), [comp]);`

  Calcula una sola componente específica de objeto tensorial. Para desplegarla en la pantalla es necesario usar `grcomponent`

- `grclear( arg );`

  Borra la memoria de un tensor calculado (no borra definiciones hechas con `grdef`, eso lo hace `grundef`). El resultado de esta instrucción depende de la variable "arg":

  arg = objeto tensorial (por ejemplo `R(up,dn)`), borra el objeto tensorial calculado en la sesión, mas no la métrica cargada.

  arg = `results`, borra todos los tensores calculados, mas no la métrica cargada.

  arg = `metric`, borra también métrica (pero no la base, en caso de estar usando una base no–coordenada).

  arg = `spacetime`, también la base.

- `grsavedef(GRTobjects, 'trayectoria' );` y `grloaddef(GRTobjects, 'trayectoria' );`

  Permite guardar en un archivo de texto (ubicado en `'trayectoria'`) definiciones de tensores hechas mediante `grdef`. Dichas definiciones se pueden cargar a una sesión de GRTensor mediante `grloaddef`.

Acknowledgments

El autor agradece apoyo financiero por parte del proyecto PAPIIT-DGAPA número IN-117803. También agradece apoyo por parte del ICN-UNAM y de la DGFM-SMF.

Apéndice

Es muy ilustrativo mostrar cómo podemos utilizar las instrucciones de manipulación y simplificación de Maple para transformar las ecuaciones de campo ($G_{ab} = \kappa T_{ab}$) y de balance ($T^{ab}{}_{;b} = 0$) en un sistema de ecuaciones diferenciales de primer orden que puede ser integrado con métodos numéricos.

---

i. El material del presente artículo está basado en el curso "Aplicaciones del cómputo simbólico a la Relatividad General", impartido por el autor en el VI Taller de la División de Gravitación y Física Matemática de la Sociedad Mexicana de Física, Metepec, Puebla, noviembre de 2005. El material del curso e información útil sobre GRTensor puede ser descargado del sitio www.smf.dgfm/VI_taller/curso_sussman.

ii. De aquíen adelante describiremos exclusivamente el uso de GRTensor en Maple, considerando que este módulo corre en Mathematica en froma completamente equivalente).

iii. En general es importante no usar nombres que coincidan con el de algún tensor, predefinido o definido por el usuario. Esta regla se aplica ya sea a objetos tensoriales que el usuario quiere definir, o a nombres de índinces, e incluso a variables usadas en la sesión de Maple. Esto descarta, por ejemplo, la posibilidad de usar nombres como: `g` (tensor métrico), `R` (tensores de Riemann y Ricci), `G` (tensor de Einstein) y `C` (tensor de Weyl). Sugerimos al lector consultar la hoja de ayuda en linea "grt_objects".

1. P. Musgrave, D. Pollney y K. Lake (1996), Queen's University, Kingston, Ontario, Canada. El software y documentación de GRTensor está disponible en el sitio www.grtensor.org
2. Waterloo Maplesoft Inc.
3. Wolfram Research Institute
4. www.smf.org.mx/dgfm/VI_taller/curso_sussman