In[1]:=

```
Print["Info : questo file è stato eseguito il ",
 DateString[], "\nSi appoggia alla seguente libreria tensoriale: \n",
 versione°libreria°data];
```

```
Info : questo file è stato eseguito il Sat 1 Mar 2014 17:16:10
Si appoggia alla seguente libreria tensoriale:
versione°libreria°data
```

Esempi di trasformazioni di tensori

Per evitare di usare male la funzione tralba[ ] della mia libreria tensoriale raccolgo qui alcuni esempi di come va usata.

La funzione tralba[ ] serve sia per fare cambiamenti di coordinate sia per cambiare il tipo di indice da COvariante ad OCvariante (uso questo termine per definire il tipo tradizionalmente chiamato controvariante)

Nel caso di un cambiamento di tipo di indice le operazioni sono meno “rischiose” perché il tensore metrico sia covariante che ocvariante è una matrice simmetrica mentre quando si fanno cambiamenti di coordinate la matrice di trasformazione (detta matrice jacobiana) è una matrice in genere NON simmetrica e dunque la matrice stessa è diversa dalla sua trasposta.

Matrici e tensori del test

Queste sono cose preliminari che mi serviranno in seguito...

In[2]:=

```
(* Mi serve per creare una matrice di trasformazione complicata *)
gcasuale[m_, mit_, fasim_] :=
  Block[{j, mm, xx, nn, md = 0}, For[j = 1, mit > j, j++,
    mm = RandomInteger[{-Abs[m], Abs[m]}, {3, 3}];
    If[fasim, mm = (Transpose[mm] + mm) / 2];
    xx = Det[mm];
    If[PrimeQ[Abs[xx]], md = Abs[xx] ;
     Print["Determinante = ", xx, " all'iterazione ", j];
     Break[]]];
   If[md == 0,
    Print["Non trovata matrice con determinante primo"]; Return[mm]];
   nn = mm;
   For[j = 1, mit > j, j++,
    mm = RandomInteger[{-Abs[m], Abs[m]}, {3, 3}]; xx = Det[mm];
    If[fasim, mm = (Transpose[mm] + mm) / 2];
    If[PrimeQ[Abs[xx]], If[md > Abs[xx], md = Abs[xx]; nn = mm];];
   Print["Valore assoluto determinante = ", md];
   Return[nn]];
```

In[3]:=

```
gg2 = gcasuale[99, 10 000, True]; Print[MatrixForm[gg2]];
Print[MatrixForm[Inverse[gg2]]];
```

```
Determinante = 1583 all'iterazione 176

Valore assoluto determinante = 107
```

$$\begin{pmatrix} 85 & 26 & \frac{73}{2} \\ 26 & 23 & \frac{27}{2} \\ \frac{73}{2} & \frac{27}{2} & 12 \end{pmatrix}$$

$$\begin{pmatrix} -\frac{375}{20\,648} & -\frac{723}{20\,648} & \frac{977}{10\,324} \\ -\frac{723}{20\,648} & \frac{1249}{20\,648} & \frac{397}{10\,324} \\ \frac{977}{10\,324} & \frac{397}{10\,324} & -\frac{1279}{5162} \end{pmatrix}$$

In[4]:=
```
tg2 = gcasuale[99, 10 000, False]; Print[MatrixForm[tg2]];
Print[MatrixForm[Inverse[tg2]]];
```

```
Determinante = -329209 all'iterazione 44

Valore assoluto determinante = 277
```

$$\begin{pmatrix} -19 & -17 & 21 \\ 32 & 58 & -61 \\ 67 & -6 & -16 \end{pmatrix}$$

$$\begin{pmatrix} \frac{1294}{277} & \frac{398}{277} & \frac{181}{277} \\ \frac{3575}{277} & \frac{1103}{277} & \frac{487}{277} \\ \frac{4078}{277} & \frac{1253}{277} & \frac{558}{277} \end{pmatrix}$$

In questo modo mi sforzo a fare una matrice di trasformazione caratterizzata da un determinante piccolo ma non troppo. Ho scelto questo metodo tra gli infiniti possibili per generare una matrice con determinante piccolo in valore assoluto ma non nullo (se venisse 0 basta dilanciare la generazione casuale della matrice)

Funzione da sperimentare :

xtralba[trasforma,tensoredacambiare,listacambiamenti]

Serve sia per fare trasformazioni di coordinate partendo da uno Jacobiano , sia a trasformare indici **co**varianti in **oc**varianti e viceversa. La correttezza di questa funzione non è facilissima da verificare dato che questa funzione gestisce enormi quantità di dati per cui è PRUDENTE fare almeno qualche verifica numerica usando numeri razionali e dunque senza introdurre errori in nessun passaggio dei calcoli.

Ricordare che la lista cambiamenti è una lista che se ha un intero positivo segnala che il cambiamento del corrispondente indice va fatto, se zero non va fatto e se negativo va fatto ma usando la trasposta dell'inversa della matrice trasforma (che puo' essere un tensore metrico o una matrice legata alla matrice jacobiana del cambiamento di coordinate voluto)

In[5]:=
```
(* tralba vuol dire TRAsformazione_ALto_BAsso *)
(* MATRICE DI TRASFORMAZIONE: *)
(* tra_ ==
 matrice di trasformazione del vecchio sistema di coordinate nel nuovo,
oppure  può essere il tensore metrico nudo usato per convertire i
```

```
 tensori da controvarianti a covarianti o viceversa dato che
 le operazioni sono pilotate dalla lista cambia *)
(* Nel caso di cambiamento degli indici consiglio di usare il tensore
 metrico COVARIANTE che serve per ottenere un vettore covariante avendo un
 vettore controvariante. Per fare questo bisogna che cambia valga {1} *)
(* Se invece è una matrice di
 trasformazione deve essere la matrice che serve
 per trasformare un vettore controvariante
 in un nuovo vettore controvariante
 nel nuovo sistema di coordinate. Dunque in questo caso tra deve essere
 l'inversa della matrice jacobiana  *)
(* Notare che bisogna avere il
 vettore delle trasformazioni per poter cambiare
 il sistema di riferimento e la matrice
 delle derivate delle funzioni detta lo
 Jacobiano, è la matrice usata, trasponendola,
per trasformare i vettori covarianti
 mentre la sua inversa serve per trasformare i vettori controvarianti *)
(* Notare che la matrice jacobiana
 NON E' SIMMETRICA per cui ovviamente anche
 la matrice inversa, di solito NON E' SIMMETRICA *)
(* QUALE TENSORE: *)
(* ten_ == tensore nudo di rango maggiore di 0 da trasformare *)
(* COSA CAMBIA: *)
(* cambia_ ==
 lista di pilotaggio del cambiamento. Se un elemento vale 1 allora
  si applica la trasformazione tra_ ,
se vale -1 allora applica la trasposta
  dell'inversa di tra_ dato che,
nel caso di trasformazione di coordinate,l'indice è covariante,
 se vale 0 non fa cambiamenti e questa opzione serve nella trasformazione
 del tipo da covariante a controvariante o viceversa *)
(* Se per convenzione tendo a
  usare solo o prevalentemente vettori covarianti
  allora per ottenere i corrispondenti controvarianti dovro' usare il
  tensore metrico covariante ed indicare con -1 quegli indici che voglio
  fare diventare controvarianti *)
xtralba[tra_, ten_, cambia_] := Block[{inTtra, i, ii, n, nc, per, nuovo},
   If[ArrayDepth[tra] ≠ 2,
    Return["Errore: primo arg non matrice ossia di tensore di rango 2"]];
   If[ArrayDepth[cambia] ≠ 1,
    Return["Errore: secondo argomento non lista"]];
   (* Se la lista cambia contiene un intero positivo,
   usa la trasformazione tra
    ma se l'intero è negativo usa l'inversa della trasformazione tra *)
   inTtra = Transpose[Inverse[tra]];
   n = ArrayDepth[ten];
   per = Range[n];
   (* Se è un vettore ossia ha rango 1 *)
   If[cambia[[1]] > 0,
    nuovo = tra.ten;,
    If[0 > cambia[[1]], nuovo = inTtra.ten;,
     nuovo = ten]];
```

```
    If[n == 1, Return[Simplify[nuovo]] ];
    (* Se invece non è un vettore fa questo... *)
    nc = Length[cambia];
    (* Se il rango del tensore è maggiore della lunghezza della lista
     continua ad usare l'ultimo elemento di cambia *)
    For[i = 2, n ≥ i, i++,
     per[[1]] = i; per[[i]] = 1;
     ii = Min[i, nc];
     If[cambia[[ii]] > 0,
      nuovo = Transpose[tra.Transpose[nuovo, per], per];,
      If[0 > cambia[[ii]],
       nuovo = Transpose[inTtra.Transpose[nuovo, per], per]]];
     per[[1]] = 1; per[[i]] = i];
    Simplify[nuovo]];
```

Ora la funzione xtralba[ ] va sperimentata...

Definisco una trasformazione di esempio. Per ottenere le coordinate sferiche tradizionali occorrono queste formule:

In[6]:=
```
(* Definisco i simboli delle coordinate sferiche in 4 dimensioni *)
esempio°sferico = {t, ρ, θ, ϕ };
esempio°incartesiano = {t, ρ Cos[θ] Cos[ϕ], ρ Cos[ θ] Sin[ ϕ], ρ Sin[θ]};
Print["Trasformazione classica da sferico in cartesiano\n",
  esempio°incartesiano];
esempio°jacocart = FullSimplify[D[esempio°incartesiano, {esempio°sferico}]];
Print["Matrice Jacobiana per cartesiana desunta:\n",
  MatrixForm[esempio°jacocart]];
```

```
Trasformazione classica da sferico in cartesiano
{t, ρ Cos[θ] Cos[ϕ], ρ Cos[θ] Sin[ϕ], ρ Sin[θ]}
```

```
Matrice Jacobiana per cartesiana desunta:
```

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \text{Cos}[\theta]\,\text{Cos}[\phi] & -\rho\,\text{Cos}[\phi]\,\text{Sin}[\theta] & -\rho\,\text{Cos}[\theta]\,\text{Sin}[\phi] \\ 0 & \text{Cos}[\theta]\,\text{Sin}[\phi] & -\rho\,\text{Sin}[\theta]\,\text{Sin}[\phi] & \rho\,\text{Cos}[\theta]\,\text{Cos}[\phi] \\ 0 & \text{Sin}[\theta] & \rho\,\text{Cos}[\theta] & 0 \end{pmatrix}$$

In[11]:=
```
(* Definisco i simboli delle coordinate cartesiane in 4 dimensioni *)
esempio°cartesiano = {t, x, y, z};
(* Definisco le funzioni di trasformazione da cartesiane in
 sferiche applicabili a patto che x non sia esattamente zero,
altrimenti uso un x positivo ma veramente trascurabile *)
esempio°insferico = {t, Sqrt[x^2 + y^2 + z^2],
   ArcSin[z / Sqrt[x^2 + y^2 + z^2]], ArcSin[y / Sqrt[x^2 + y^2]]};
Print["Trasformazione classica da cartesiano in sferico\n",
  esempio°insferico];
esempio°jacosfer = FullSimplify[D[esempio°insferico, {esempio°cartesiano}]];
Print["Matrice Jacobiana per sferica desunta:\n",
 MatrixForm[esempio°jacosfer]];
```

Trasformazione classica da cartesiano in sferico

$$\left\{t,\ \sqrt{x^2+y^2+z^2}\ ,\ \text{ArcSin}\left[\frac{z}{\sqrt{x^2+y^2+z^2}}\right],\ \text{ArcSin}\left[\frac{y}{\sqrt{x^2+y^2}}\right]\right\}$$

Matrice Jacobiana per sferica desunta:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{x}{\sqrt{x^2+y^2+z^2}} & \frac{y}{\sqrt{x^2+y^2+z^2}} & \frac{z}{\sqrt{x^2+y^2+z^2}} \\ 0 & -\frac{x\,z}{\sqrt{\frac{x^2+y^2}{x^2+y^2+z^2}}\ \left(x^2+y^2+z^2\right)^{3/2}} & -\frac{y\,z}{\sqrt{\frac{x^2+y^2}{x^2+y^2+z^2}}\ \left(x^2+y^2+z^2\right)^{3/2}} & \frac{\sqrt{\frac{x^2+y^2}{x^2+y^2+z^2}}}{\sqrt{x^2+y^2+z^2}} \\ 0 & -\frac{x\,y}{\sqrt{\frac{x^2}{x^2+y^2}}\ \left(x^2+y^2\right)^{3/2}} & \frac{\sqrt{\frac{x^2}{x^2+y^2}}}{\sqrt{x^2+y^2}} & 0 \end{pmatrix}$$

Quando debbo trasformare un tensore ocvariante...

In[15]:=
```
esempio°gsfe211 =
  xtralba[Inverse[esempio°jacocart], DiagonalMatrix[{1, -1, -1, -1}], {1}];
Print["Tensore metrico ocvariante in coordinate sferiche\n",
  MatrixForm[esempio°gsfe211]];
Print["Oppure posso fare così \n", MatrixForm[
   xtralba[Transpose[esempio°jacocart], DiagonalMatrix[{1, -1, -1, -1}], {-1}]]];
```

Tensore metrico ocvariante in coordinate sferiche

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -\frac{1}{\rho^2} & 0 \\ 0 & 0 & 0 & -\frac{\text{Sec}[\theta]^2}{\rho^2} \end{pmatrix}$$

Oppure posso fare così

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -\frac{1}{\rho^2} & 0 \\ 0 & 0 & 0 & -\frac{\text{Sec}[\theta]^2}{\rho^2} \end{pmatrix}$$

Quando devo trasformare un tensore covariante...

In[18]:=
```
esempio°gsfe2 =
  xtralba[Transpose[esempio°jacocart], DiagonalMatrix[{1, -1, -1, -1}], {1}];
Print["Tensore metrico covariante in coordinate sferiche\n",
  MatrixForm[esempio°gsfe2]];
Print["Oppure posso fare così \n", MatrixForm[
   xtralba[Inverse[esempio°jacocart], DiagonalMatrix[{1, -1, -1, -1}], {-1}]]];
```

Tensore metrico covariante in coordinate sferiche

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -\rho^2 & 0 \\ 0 & 0 & 0 & -\rho^2 \text{Cos}[\theta]^2 \end{pmatrix}$$

Oppure posso fare così

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -\rho^2 & 0 \\ 0 & 0 & 0 & -\rho^2 \text{Cos}[\theta]^2 \end{pmatrix}$$

Ora che ho ottenuto il tensore metrico in coordinate sferiche provo ad ottenere, da questo, il tensore metrico cartesiano (che, eccezionalmente, coincide col suo inverso ossia il covariante coincide con l'ocvariante).

In[21]:=
```
sostituito211 = Simplify[esempio°gsfe211 /. ρ → esempio°insferico[[2]] /.
    θ → esempio°insferico[[3]]];
Print["Innanzi tutto esprimo le sferiche come cartesiane \n",
  MatrixForm[sostituito211],
  "\npoi..."];
esempio°gcart211 = xtralba[Inverse[esempio°jacosfer], sostituito211, {1}];
Print["Tensore metrico ocvariante in coordinate cartesiane\n",
  MatrixForm[esempio°gcart211]];
Print["Oppure posso fare così operando su un tensore totalmente covariante"];
sostituito2 = Simplify[esempio°gsfe2 /. ρ → esempio°insferico[[2]] /.
    θ → esempio°insferico[[3]]];
Print["Innanzi tutto esprimo le sferiche come cartesiane \n",
  MatrixForm[sostituito2],
  "\npoi..."];
Print["E poi applico xtralba \n", MatrixForm[
   xtralba[Transpose[esempio°jacosfer], sostituito2, {1}]]];
```

Innanzi tutto esprimo le sferiche come cartesiane

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -\frac{1}{x^2+y^2+z^2} & 0 \\ 0 & 0 & 0 & -\frac{1}{x^2+y^2} \end{pmatrix}$$

poi...

Tensore metrico ocvariante in coordinate cartesiane

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$$

Oppure posso fare così operando su un tensore totalmente covariante

Innanzi tutto esprimo le sferiche come cartesiane

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -x^2-y^2-z^2 & 0 \\ 0 & 0 & 0 & -x^2-y^2 \end{pmatrix}$$

poi...

E poi applico xtralba

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$$

Questo esempio dimostra che la xtralba funziona bene nel fare cambiamenti di coordinate. Naturalmente la sua utilità consiste nel poter trattare anche tensori di rango superiore ad 1 ossia matrici, vettori di matrici etc...

Esempi vari di trasformazioni tra indici co ed oc.

In queste trasformazioni la caratteristica condivisa è il fatto che la matrice di trasformazione, ovvero il tensore metrico, è una matrice SIMMETRICA il che potrebbe mascherare errori nelle operazioni fatte dalla funzione xtralba[ ] ma ovviamente la funzione xtralba[ ] è di uso comodo quando si devono fare contrazioni tra tensori di rango maggiore ad 1 ossia dalle matrici in sù, viceversa le contrazioni tra vettori sono semplici e largamente note alla gente ossia fattibili usando le normali funzioni di *Mathematica*.

Usando vettori ossia tensori di rango 1

Creo due vettori a caso e ipotizzo che il tensore metrico covariante sia gg2. Lavoro, per comodità, solo in spazi di tre dimensioni.

In[29]:=
```
va1 = RandomInteger[{-9, 99}, {3}]; vb1 = RandomInteger[{-9, 99}, {3}]
```

Out[29]=
```
{84, 89, 25}
```

In[30]:=
```
scalare = va1.Inverse[gg2].vb1
```

Out[30]=
$$\frac{9\,372\,421}{20\,648}$$

Trasforma anticipatamente il vettore covariante vb1 nel suo ocvariante vb11:

In[31]:=
```
vb11 = xtralba[gg2, vb1, {-1}]
```

Out[31]=
$$\left\{-\frac{46\,997}{20\,648}, \frac{70\,279}{20\,648}, \frac{53\,451}{10\,324}\right\}$$

In[32]:=
```
Print[va1.vb11, " deve valere 0 : ", va1.vb11 - scalare]
```

$$\frac{9\,372\,421}{20\,648}\ \text{deve valere 0 : 0}$$

Trasforma anticipatamente il vettore covariante va1 nel suo ocvariante va11:

In[33]:=
```
va11 = xtralba[gg2, va1, {-1}]
```

Out[33]=
$$\left\{\frac{167\,639}{20\,648}, \frac{69\,155}{20\,648}, -\frac{217\,281}{10\,324}\right\}$$

In[34]:=
```
{va11.vb1, va11.vb1 - scalare}
```

Out[34]=
$$\left\{\frac{9\,372\,421}{20\,648}, 0\right\}$$

Usando tensori di rango 2

Operare con tensori di rango 2 ossia le consuete matrici quadrate dell'algebra lineare è già un po' più complicato.
Lavoro sempre in spazi tridimensionali usando come tensore metrico covariante il tensore gg2.

In[35]:=
```
ma2 = RandomInteger[{-9, 99}, {3, 3}]; mb2 = RandomInteger[{-9, 99}, {3, 3}];
baseab = ma2.Inverse[gg2].mb2; Print[MatrixForm[baseab]];
```

$$\begin{pmatrix} \frac{639\,650}{2581} & \frac{18\,769}{116} & \frac{3\,637\,283}{20\,648} \\ \frac{100\,876}{2581} & \frac{3901}{58} & \frac{1\,854\,039}{10\,324} \\ \frac{901\,672}{2581} & \frac{51\,061}{116} & \frac{17\,197\,375}{20\,648} \end{pmatrix}$$

Trasforma anticipatamente il primo indice del secondo tensore di rango 2

In[37]:=
```
mb21 = xtralba[gg2, mb2, {-1, 0}]; Print[MatrixForm[mb21]];
Print[MatrixForm[ma2.mb21 - baseab]];
```

$$\begin{pmatrix} \frac{1008}{2581} & \frac{159}{116} & \frac{106\,013}{20\,648} \\ \frac{9418}{2581} & \frac{507}{116} & \frac{130\,721}{20\,648} \\ -\frac{2692}{2581} & -\frac{353}{58} & -\frac{153\,027}{10\,324} \end{pmatrix}$$

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Trasforma anticipatamente il secondo indice del primo tensore di rango 2

In[38]:=
```
ma221 = xtralba[gg2, ma2, {0, -1}]; Print[MatrixForm[ma221]];
Print[MatrixForm[ma221.mb2 - baseab]];
```

$$\begin{pmatrix} \frac{1223}{20\,648} & \frac{82\,307}{20\,648} & -\frac{15\,465}{10\,324} \\ -\frac{7165}{10\,324} & -\frac{4729}{10\,324} & \frac{13\,987}{5162} \\ -\frac{74\,701}{20\,648} & \frac{61\,135}{20\,648} & \frac{88\,683}{10\,324} \end{pmatrix}$$

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Quanto segue è di meno frequente utilizzo ma non meno importante per certe applicazioni del

calcolo tensoriale...

Scalare totale di un tensore di rango 2 (utile per fare l'invariante del campo elettromagnetico).

Per esempio quando uno vuol fare questo calcolo:

$$F^{h,k}F_{h,k}$$

Ovviamente Il tensore elettromagnetico è un tensore antisimmetrico ma qui in generale per un tensore di rango 2...

In[39]:=
```
(* Invariante
 totale : applicabile a tensori di qualsiasi rango in spazi di
   qualsiasi numero di dimensioni *)
xfainvaTotale[tco_, g2_] := Block[{toc, toc°tco, ade},
   ade = ArrayDepth[tco];
   toc = xtralba[g2, tco, {-1}];
   toc°tco = TensorProduct[toc, tco];
   Simplify[TensorContract[toc°tco, Array[{#1, #1 + ade} &, {ade}]]]];
(* Si tratta di una funzione potentissima che, mio parere,
 va molto sperimentata perché si basa su funzioni di Mathematica che,
sperabilmente, sono corrette ma... ragionando da S. Tommaso... *)
```

Per esempio, prendendo una qualsiasi matrice covariante ossia un tensore di rango 2 covariante qui dovrebbe venir fuori lo stesso risultato:

In[40]:=
```
Print[Tr[Transpose[Inverse[gg2].(ma2.Inverse[gg2])].ma2],
  " oppure ", xfainvaTotale[ma2, gg2]];
```

$$\frac{16\,022\,734\,329}{426\,339\,904}\ \text{oppure}\ \frac{16\,022\,734\,329}{426\,339\,904}$$

La funzione xfainvaTotale[ ] funziona **anche** con tensori di rango 1 ossia vettori:

In[41]:=
```
vet1 = RandomInteger[{-99, 999}, {3}];
Print[vet1.Inverse[gg2].vet1, " oppure ", xfainvaTotale[vet1, gg2]];
```

$$-\frac{646\,184\,551}{20\,648}\ \text{oppure}\ -\frac{646\,184\,551}{20\,648}$$

In[42]:=
```
grn2 = gsfern222; Print[MatrixForm[grn2]];
```

```
gsfern222
```

Il tensore di Riemann del tensore metrico covariante grn2

In[43]:=
```
Rrn4 = faRiemann42222[grn2, {t, r, h, p}];
```

In[44]:=
```
invarn0 = xfainvaTotale[Rrn4, grn2];
```

Ecco l'invariante totale quadratico della metrica di Reissner-Nordström in coordinate sferiche. Per verifica guardare sul libro di Hobson, Efstathiou e Lasenby (ISBN 9780521829519) a pag. 250. Si parla del KRETSCHMANN SCALAR di cui parla diffusamente anche RICHARD CONN HENRY a proposito della "Kerr-Newman Black Hole".

In[45]:=
```
invarn0
```

Out[45]=
```
TensorContract[Errore: primo arg non matrice ossia di tensore di rango 2⊗
  faRiemann42222[gsfern222, {t, r, h, p}], {{1, 2}}]
```

http://arxiv.org/abs/astro-ph/9912320

http://iopscience.iop.org/0004-637X/535/1/350/

Usando tensori di rango 3 e superiori

Qui faccio alcune varianti della stessa funzione ma tali da utilizzare in vario modo la funzione che voglio mettere alla prova.

In[46]:=
```
(* Mi limito a fare invarianti totali solo di tensori di rango 3 *)
xfainvaTotale3[tco_, g2_] := Block[{toc, toc°tco, ade},
   ade = ArrayDepth[tco];
   If[ade ≠ 3, Print["Rango errato = ", ade,
     "\nQuesta funzione tratta solo tensori covarianti di  rango 3"];
    Return[False]];
   toc = xtralba[g2, tco, {-1, -1, -1}];
   toc°tco = TensorProduct[toc, tco];
   Simplify[TensorContract[toc°tco, {{1, 4}, {2, 5}, {3, 6}} ]]];
(* oppure ... *)
yfainvaTotale3[tco_, g2_] := Block[{g2oc, toc, toc°tco, ade},
   ade = ArrayDepth[tco];
   If[ade ≠ 3, Print["Rango errato = ", ade,
     "\nQuesta funzione tratta solo tensori covarianti di  rango 3"];
    Return[False]];
   g2oc = Inverse[g2];
   toc = xtralba[g2oc, tco, {1, 1, 1}];
   toc°tco = TensorProduct[toc, tco];
   Simplify[TensorContract[toc°tco, {{1, 4}, {2, 5}, {3, 6}} ]]];
```

In[48]:=
```
teco3 = RandomInteger[{-99, 999}, {3, 3, 3}];
Print["Dato questo tensore di rango 3\n", MatrixForm[teco3]];
Print[xfainvaTotale[teco3, gg2], " oppure ", xfainvaTotale3[teco3, gg2],
  " oppure ", yfainvaTotale3[teco3, gg2]];
```

Dato questo tensore di rango 3

$$\begin{pmatrix} \begin{pmatrix} 325 \\ -92 \\ 65 \end{pmatrix} & \begin{pmatrix} 468 \\ 816 \\ 369 \end{pmatrix} & \begin{pmatrix} 28 \\ 278 \\ 71 \end{pmatrix} \\ \begin{pmatrix} 18 \\ 53 \\ 307 \end{pmatrix} & \begin{pmatrix} -86 \\ 540 \\ 659 \end{pmatrix} & \begin{pmatrix} 711 \\ 568 \\ 537 \end{pmatrix} \\ \begin{pmatrix} 67 \\ 780 \\ 341 \end{pmatrix} & \begin{pmatrix} 710 \\ 789 \\ 478 \end{pmatrix} & \begin{pmatrix} 972 \\ 709 \\ 816 \end{pmatrix} \end{pmatrix}$$

$$-\frac{255\,344\,932\,917\,873}{275\,095\,823\,056}\ \text{oppure}\ -\frac{255\,344\,932\,917\,873}{275\,095\,823\,056}\ \text{oppure}\ -\frac{255\,344\,932\,917\,873}{275\,095\,823\,056}$$

In[51]:=
```
(* Qui ripeto gli esperimenti ma solo per tensori di rango 4... come lo
 è l'importantissimo tensore di Riemann *)
xfainvaTotale4[tco_, g2_] := Block[{toc, toc°tco, ade},
   ade = ArrayDepth[tco];
   If[ade ≠ 4, Print["Rango errato = ", ade,
     "\nQuesta funzione tratta solo tensori covarianti di  rango 4"];
    Return[False]];
   toc = xtralba[g2, tco, {-1, -1, -1, -1}];
   toc°tco = TensorProduct[toc, tco];
   Simplify[TensorContract[toc°tco, {{1, 5}, {2, 6}, {3, 7}, {4, 8}} ]]];
(* Oppure *)
yfainvaTotale4[tco_, g2_] := Block[{g2oc, toc, toc°tco, ade},
   ade = ArrayDepth[tco];
   If[ade ≠ 4, Print["Rango errato = ", ade,
     "\nQuesta funzione tratta solo tensori covarianti di  rango 4"];
    Return[False]];
   g2oc = Inverse[g2];
   toc = xtralba[g2oc, tco, {1, 1, 1, 1}];
   toc°tco = TensorProduct[toc, tco];
   Simplify[TensorContract[toc°tco, {{1, 5}, {2, 6}, {3, 7}, {4, 8}} ]]];
```

In[53]:=
```
teco4 = RandomInteger[{-99, 999}, {3, 3, 3, 3}];
Print["Dato questo tensore di rango 4\n", MatrixForm[teco4]];
Print[xfainvaTotale[teco4, gg2], " oppure ", xfainvaTotale4[teco4, gg2],
  " oppure ", yfainvaTotale4[teco4, gg2]];
```

```
Dato questo tensore di rango 4
```

$$\begin{pmatrix} \begin{pmatrix} 820 & 54 & 61 \\ 418 & 56 & 255 \\ -2 & 254 & 2 \end{pmatrix} & \begin{pmatrix} 697 & 632 & 524 \\ 759 & -14 & 558 \\ 212 & -20 & 661 \end{pmatrix} & \begin{pmatrix} 261 & 90 & 777 \\ 837 & 749 & 902 \\ 547 & 374 & 749 \end{pmatrix} \\ \begin{pmatrix} 659 & 146 & 642 \\ 254 & 471 & -51 \\ 536 & 662 & 447 \end{pmatrix} & \begin{pmatrix} 98 & 863 & 38 \\ 893 & 386 & 981 \\ 547 & 879 & 772 \end{pmatrix} & \begin{pmatrix} 617 & 957 & 868 \\ 128 & -83 & 78 \\ 707 & 906 & 981 \end{pmatrix} \\ \begin{pmatrix} 451 & -19 & 670 \\ 597 & 166 & 552 \\ 774 & 196 & 506 \end{pmatrix} & \begin{pmatrix} 831 & 119 & 254 \\ 538 & 376 & 572 \\ 101 & 157 & 197 \end{pmatrix} & \begin{pmatrix} -48 & 948 & 181 \\ -25 & 906 & 62 \\ 459 & 196 & -24 \end{pmatrix} \end{pmatrix}$$

$$-\frac{1\,541\,395\,290\,249\,521\,967}{45\,441\,428\,435\,682\,304} \text{ oppure } -\frac{1\,541\,395\,290\,249\,521\,967}{45\,441\,428\,435\,682\,304} \text{ oppure } -\frac{1\,541\,395\,290\,249\,521\,967}{45\,441\,428\,435\,682\,304}$$

Con questo mi sembra di avere dimostrato che è ragionevole fidarsi dei risultati della funzione xfainvaTotale capace di trattare tensori di qualsiasi rango maggiore di 0.

Esempi di vere trasformazioni di coordinate

Tratto solo trasformazioni notissime ossia quelle dalle cartesiane bi o tridimensionali alle polari o alle sferiche.

Studio di trasformazioni polari e sferiche incluse varianti “algebriche” poco usate ma interessanti.

Esaminiamo un caso relativamente facile. Per le coordinate cilindriche, alias polari, ossia un problema bidimensionale:

In[56]:=
```
(* Normalizza il valore in modo che stia sicuramente tra 0 e 2 Pi *)
cilnorm[c_] := {c[[1]], Mod[c[[2]], 2 Pi]};
(* Da cilindriche a cartesiane: formula semplicissima  *)
cil2c[c_] := Simplify[ c[[1]] {Cos[c[[2]]], Sin[c[[2]]]}];
(* Da cartesiane a cilindriche: formula piuttosto delicata *)
c2cil[c_] := Block[{p, rr = Sqrt[c[[1]]^2 + c[[2]]^2]},
   If[rr > 0, p = 2 ArcTan[Abs[c[[2]]] / (rr + Abs[c[[1]]])];
    If[0 > c[[1]], p = Pi - p]; If[0 > c[[2]], p = 2 Pi - p], p = 0];
   {rr, p}];
(* per fare la verifica bisogna
 sempre normalizzare le coordinate cilindriche
 in modo che la longitudine vada sempre da 0 a 2 Pi *)
serro = 0; For[j = 0, 100 > j, j++, cil = {1, -13 + j / 3};
 errore = Abs[ N[N[cilnorm[cil], 16] - N[c2cil[cil2c[cil]], 16] ]];
 serro += errore[[2]];
 If[errore[[2]] > 1 / 10^12,
  Print[cil2c[cil], " ", errore, "  Strano..."]]];
Print["Ha fatto: ", j, " casi. Somma errori ", serro];
```

```
Ha fatto: 100 casi. Somma errori 0.
```

Questo caso mi è servito per dare una robusta definizione al caso seguente, di interesse più generale.

Ora tratto il problema più difficile ossia lavoro in 3 dimensioni passando dalle coordinate sferiche tradizionali a quelle cartesiane e viceversa.
Il problema è da trattare con cura per gestire correttamente anche i casi non normalizzati ossia quando si hanno angoli negativi o quando la latitudine supera Pi e la longitudine supera 2 Pi.

In[61]:=
```
(* trasformazione classica da sferiche a cartesiane ,
con uso della latitudine
 come seconda coordinata e della longitudine come terza *)
(* Notare che la latitudine deve andare da 0 a Pi e non da Pi/2 a -
 Pi/2 come si usa in geografia ossia
  il polo nord ha latitudine 0 e il polo sud ha latitudine Pi *)
(* Analogamente la longitudine deve andare da 0 a 2 Pi *)
(* Normalizza gli angoli delle coordinate sferiche *)
sfenorm[s_] := Block[{h = s[[2]], p = s[[3]]},
   If[0 > h, h = Mod[-h, Pi]; p = Pi + p, If[h > Pi, h = Mod[h, Pi];
     p = Pi + p]]; {s[[1]], h, Mod[p, 2 Pi]}];
(* Da sferiche a cartesiane *)
s2c[s_] := Block[{h = s[[2]], p = s[[3]], sh, ch},
   If[0 > h, h = Mod[-h, Pi]; p = Pi + p, If[h > Pi, h = Mod[h, Pi];
     p = Pi + p]]; p = Mod[p, 2 Pi];
   sh = Sin[h]; ch = Cos[h];
   Simplify[s[[1]] { sh Cos[p], sh Sin[p], ch}]];
(* Da cartesiane a sferiche normalizzate *)
c2s[c_] := Block[{ rq = c[[1]]^2 + c[[2]]^2, rr, rz, p, h},
   rz = Sqrt[rq + c[[3]]^2]; rr = Sqrt[rq];
   (* c[[1]] è la x e c[[2]] è la y *)
   If[rr > 0, p = 2 ArcTan[Abs[c[[2]]] / (rr + Abs[c[[1]]])];
    If[0 > c[[1]], p = Pi - p]; If[0 > c[[2]], p = 2 Pi - p], p = 0];
   (* c[[3]] è la z *)
   If[rz > 0, h = 2 ArcTan[rr / (rz + Abs[c[[3]]])];
    If[0 > c[[3]], h = Pi - h], h = 0];
   Simplify[{rz, h, p}]] ;
(* Mi sembra che questo funzioni... *)
```

La trasformazione inversa, ossia il calcolo delle coordinate sferiche date quelle cartesiane, non è una cosa banalissima perché bisogna evitare il rischio di divisioni per zero.
Verifico attentamente il corretto funzionamento delle formule sia in casi specifici sia in casi generati a caso:

Ecco una rapida verifica..

In[64]:=
```
psfe = {121/100, -223/100, 120/75}; Print[psfe]; incar = s2c[psfe];
psfen = c2s[incar]; N[N[psfen, 24] - N[sfenorm[psfe], 24], 24]
```

$$\left\{\frac{121}{100}, -\frac{223}{100}, \frac{8}{5}\right\}$$

Out[64]= $\{0.\times 10^{-24}, 0.\times 10^{-24}, 0.\times 10^{-24}\}$

Una serie di prove a caso...

In[65]:=
```
serro = 0; For[j = 0, 30 > j, j++,
 psfe = RandomInteger[{-999, 999}, {3}] / 100; psfe[[1]] = 1 + Abs[psfe[[1]]];
 (* Qui va in cartesiano e torna in sferiche normalizzate *)
 incar = s2c[psfe]; psfen = c2s[incar];
 errore = Abs[ N[psfen, 24] - N[sfenorm[psfe], 24]];
 serro += errore[[2]] + errore[[3]];
 If[(errore[[2]] + errore[[3]]) > 1 / 10^12,
  Print[psfe[cil], " ", errore]]];
Print["Eseguiti= ", j, " Ultimo= ", psfe, " Somma errori cumulati= ", serro];
(* Si spera che la somma del valore assoluto degli errori sia trascurabile *)
```

Eseguiti= 30 Ultimo= $\left\{\frac{311}{100}, \frac{431}{100}, -\frac{637}{100}\right\}$ Somma errori cumulati= $0. \times 10^{-22}$

Una singola prova con molte stampe...

In[67]:=
```
psfe = {1, 110 / 1000, 1871 / 1000}; Print[psfe]; Print[sfenorm[psfe]]
incar = s2c[psfe]; psfen = c2s[incar];
Print[" Punto in coordinate sferiche: ", psfe,
  "\n Ossia, numerico: ", N[psfe, 24],
  "\n Normalizzato: ", sfenorm[psfe],
  "\n Cartesiano: ", incar,
  "\n Ossia, numerico: ", N[incar, 24],
  "\n Ricalcolato sferico ", psfen,
  "\n Ossia, numerico: ", N[psfen, 24],
  "\n Ricalcolandolo dovrei ottenere una differenza nulla: ",
  N[N[psfen, 24] - N[sfenorm[psfe], 24], 24]];
```

$$\left\{1, \frac{11}{100}, \frac{1871}{1000}\right\}$$

$$\left\{1, \frac{11}{100}, \frac{1871}{1000}\right\}$$

Punto in coordinate sferiche: $\left\{1, \frac{11}{100}, \frac{1871}{1000}\right\}$

Ossia, numerico:
{1.0000000000000000000000, 0.11000000000000000000000, 1.87100000000000000000000}

Normalizzato: $\left\{1, \frac{11}{100}, \frac{1871}{1000}\right\}$

Cartesiano: $\left\{\text{Cos}\left[\frac{1871}{1000}\right] \text{Sin}\left[\frac{11}{100}\right], \text{Sin}\left[\frac{11}{100}\right] \text{Sin}\left[\frac{1871}{1000}\right], \text{Cos}\left[\frac{11}{100}\right]\right\}$

Ossia, numerico: {-0.03246306574878021132008 47,
0.10486860682247871073183 8, 0.99395609795669685035784 0}

Ricalcolato sferico $\left\{\sqrt{\text{Cos}\left[\frac{11}{100}\right]^2 + \text{Sin}\left[\frac{11}{100}\right]^2 \left(\text{Cos}\left[\frac{1871}{1000}\right]^2 + \text{Sin}\left[\frac{1871}{1000}\right]^2\right)}\right.$,

$$2\,\text{ArcTan}\left[\frac{\text{Sin}\left[\frac{11}{100}\right]\sqrt{\text{Cos}\left[\frac{1871}{1000}\right]^2 + \text{Sin}\left[\frac{1871}{1000}\right]^2}}{\text{Cos}\left[\frac{11}{100}\right] + \sqrt{\text{Cos}\left[\frac{11}{100}\right]^2 + \text{Sin}\left[\frac{11}{100}\right]^2 \left(\text{Cos}\left[\frac{1871}{1000}\right]^2 + \text{Sin}\left[\frac{1871}{1000}\right]^2\right)}}\right],$$

$$\left.\pi + 2\,\text{ArcTan}\left[\frac{\text{Sin}\left[\frac{1871}{1000}\right]}{\text{Cos}\left[\frac{1871}{1000}\right] - \sqrt{\text{Cos}\left[\frac{1871}{1000}\right]^2 + \text{Sin}\left[\frac{1871}{1000}\right]^2}}\right]\right\}$$

Ossia, numerico: {1.0000000000000000000000,
0.11000000000000000000000, 1.87100000000000000000000}
Ricalcolandolo dovrei ottenere una differenza nulla: $\{0. \times 10^{-24}, 0. \times 10^{-25}, 0. \times 10^{-24}\}$

La critica a questa trasformazione + antitrasformazione è che NON è algebrica dato che richiede l'uso di funzioni trigonometriche e dunque i valori delle coordinate cartesiane non sono, in genere, numeri razionali ossia rapporti tra numeri interi, ma sono numeri trascendenti.

Viceversa è possibile usare trasformazioni di coordinate da cartesiane a SFERICHE ALGEBRICHE BASATE SULLE TANGENTI DI LATITUDINE E LONGITUDINE e ovviamente al contrario dalle sferiche algebriche alle cartesiane.

Con questo tipo di coordinate, se si parte da coordinate sferiche algebriche definite come numeri razionali, si ottengono coordinate cartesiane costituite anche loro da numeri razionali per cui, ovviamente, essendo la trasformazione reversibile, dalle coordinate cartesiane razionali si ottengono coordinate sferiche razionali.

In[70]:=
```
(* Occorre questa normalizzazione nei
 valori limiti ossia 1 in valore assoluto *)
snor[s_] := Block[{th = s[[2]], tt = s[[3]]},
   If[Abs[th] == 1, tt = 0];
   (* tangenti esterne all'intervallo -1 ... 1 *)
   If[th > 1, th = 1 / th; If[tt == -1, tt = 0, tt = (tt - 1) / (1 + tt)]];
   If[-1 > th, th = 1 / th; If[tt == -1, tt = 0, tt = (tt - 1) / (1 + tt)]];
   If[-1 ≥ tt , tt = -1 / tt];
   If[tt > 1, tt = -1 / tt]; {s[[1]], th, tt}];
(* Da sferiche algebriche ossia usando tangenti... a cartesiane
 Sono tangenti s[[2]] e s[[3]] *)
sn2c[s_] := Block[{th, tq, sm, cm, snp, cnp, snh, cnh},
   (* per la latitudine algebrica *)
   th = s[[2]]^2; snh = 2 s[[2]] / (1 + th); cnh = (1 - th) / (1 + th);
   (* per la longitudine algebrica *)
   tq = s[[3]]^2; sm = 2 s[[3]] / (1 + tq);
   cm = (1 - tq) / (1 + tq); snp = 2 sm cm; cnp = cm^2 - sm^2;
   s[[1]] {cnh cnp, cnh snp, snh}];
(* da cartesiane a sferiche algebriche *)
cn2s[c_] :=
  Block[{ x = c[[1]], y = c[[2]], z = c[[3]], rq, rr, rz, rb, tp = 0, th = 0},
   rq = x^2 + y^2;
   rz = Sqrt[rq + z^2];
   If[rz > 0, rr = Sqrt[rq]; th = z / (rr + rz);
    rb = Sqrt[2 rq + 2 Abs[x] rr];
    If[rb > 0,
     If[x > 0, tp = y / (x + rr + rb) ,
      If[y == 0, tp = 1, tp = Sign[y] (Abs[x] + rr) / (Abs[y] + rb)] ]]];
   {rz, th, tp}];
(* Le sferiche algebriche sono
  legate alla tangente di un mezzo della latitudine
  e alla tangente di un quarto della longitudine per cui l'
  intervallo fondamentale delle due tangenti è tra -1 e +1 *)
```

Assegno { r,Tan[h/2],Tan[p/4] } con il vincolo che il secondo e terzo coefficiente non sia maggiore di 1 in valore assoluto.

In[73]:=
```
ss = {1, 0, 11 / 10}; cc = sn2c[ss]; dd = cn2s[cc] - snor[ss];
Print[ss, " cc= ", cc, " ritrova= ", cn2s[cc], " errore= ", dd ];
```

$\left\{1, 0, \frac{11}{10}\right\}$ cc= $\left\{-\frac{47\,959}{48\,841}, -\frac{9240}{48\,841}, 0\right\}$ ritrova= $\left\{1, 0, -\frac{10}{11}\right\}$ errore= {0, 0, 0}

Casi speciali:

In[75]:=
```
ss = {1, 1, 1}; cc = sn2c[ss]; dd = cn2s[cc] - snor[ss];
Print[ss, " cc= ", cc, " ritrova= ", cn2s[cc], " errore= ", dd];
ss = {1, 1, 0}; cc = sn2c[ss]; dd = cn2s[cc] - snor[ss];
Print[ss, " cc= ", cc, " ritrova= ", cn2s[cc], " errore= ", dd];
ss = {1, 1, -1}; cc = sn2c[ss]; dd = cn2s[cc] - snor[ss];
Print[ss, " cc= ", cc, " ritrova= ", cn2s[cc], " errore= ", dd];
ss = {1, 0, 1}; cc = sn2c[ss]; dd = cn2s[cc] - snor[ss];
Print[ss, " cc= ", cc, " ritrova= ", cn2s[cc], " errore= ", dd];
ss = {1, 0, 0}; cc = sn2c[ss]; dd = cn2s[cc] - snor[ss];
Print[ss, " cc= ", cc, " ritrova= ", cn2s[cc], " errore= ", dd];
ss = {1, 0, -1}; cc = sn2c[ss]; dd = cn2s[cc] - snor[ss];
Print[ss, " cc= ", cc, " ritrova= ", cn2s[cc], " errore= ", dd];
ss = {1, -1, 1}; cc = sn2c[ss]; dd = cn2s[cc] - snor[ss];
Print[ss, " cc= ", cc, " ritrova= ", cn2s[cc], " errore= ", dd];
ss = {1, -1, 0}; cc = sn2c[ss]; dd = cn2s[cc] - snor[ss];
Print[ss, " cc= ", cc, " ritrova= ", cn2s[cc], " errore= ", dd];
ss = {1, -1, -1}; cc = sn2c[ss]; dd = cn2s[cc] - snor[ss];
Print[ss, " cc= ", cc, " ritrova= ", cn2s[cc], " errore= ", dd];
```

```
{1, 1, 1} cc= {0, 0, 1} ritrova= {1, 1, 0} errore= {0, 0, 0}

{1, 1, 0} cc= {0, 0, 1} ritrova= {1, 1, 0} errore= {0, 0, 0}

{1, 1, -1} cc= {0, 0, 1} ritrova= {1, 1, 0} errore= {0, 0, 0}

{1, 0, 1} cc= {-1, 0, 0} ritrova= {1, 0, 1} errore= {0, 0, 0}

{1, 0, 0} cc= {1, 0, 0} ritrova= {1, 0, 0} errore= {0, 0, 0}

{1, 0, -1} cc= {-1, 0, 0} ritrova= {1, 0, 1} errore= {0, 0, 0}

{1, -1, 1} cc= {0, 0, -1} ritrova= {1, -1, 0} errore= {0, 0, 0}

{1, -1, 0} cc= {0, 0, -1} ritrova= {1, -1, 0} errore= {0, 0, 0}

{1, -1, -1} cc= {0, 0, -1} ritrova= {1, -1, 0} errore= {0, 0, 0}
```

Qualche caso scelto a piacere...

In[93]:=
```
ss = {1, -1, -1}; cc = sn2c[ss]; dd = cn2s[cc] - snor[ss];
Print[ss, " cc= ", cc, " ritrova= ", cn2s[cc], " errore= ", dd];
```

```
{1, -1, -1} cc= {0, 0, -1} ritrova= {1, -1, 0} errore= {0, 0, 0}
```

In[95]:=
```
ss = {1, 47 / 100, -1}; cc = sn2c[ss]; dd = cn2s[cc] - snor[ss];
Print[ss, " cc= ", cc, " ritrova= ", cn2s[cc], " errore= ", dd];
```

$$\left\{1, \frac{47}{100}, -1\right\} \text{ cc= } \left\{-\frac{7791}{12\,209}, 0, \frac{9400}{12\,209}\right\} \text{ ritrova= } \left\{1, \frac{47}{100}, 1\right\} \text{ errore= } \{0, 0, 0\}$$

In[97]:=
```
ss = {1, 37 / 50, 1}; cc = sn2c[ss]; dd = cn2s[cc] - snor[ss];
Print[ss, " cc= ", cc, " ritrova= ", cn2s[cc], " errore= ", dd];
```

$$\left\{1, \frac{37}{50}, 1\right\} \text{ cc= } \left\{-\frac{1131}{3869}, 0, \frac{3700}{3869}\right\} \text{ ritrova= } \left\{1, \frac{37}{50}, 1\right\} \text{ errore= } \{0, 0, 0\}$$

In[99]:=
```
ss = {1, 1, -1 / 3}; cc = sn2c[ss]; dd = cn2s[cc] - snor[ss];
Print[ss, " cc= ", cc, " ritrova= ", cn2s[cc], " errore= ", dd];
```

$$\left\{1, 1, -\frac{1}{3}\right\} \text{ cc= } \{0, 0, 1\} \text{ ritrova= } \{1, 1, 0\} \text{ errore= } \{0, 0, 0\}$$

In[101]:=
```
ss = {1, 999 / 1000, 1 / 3}; cc = sn2c[ss]; dd = cn2s[cc] - ss;
Print[ss, " cc= ", cc, " ritrova= ", cn2s[cc], " errore= ", dd ];
```

$\left\{1, \frac{999}{1000}, \frac{1}{3}\right\}$ cc= $\left\{\frac{13\,993}{49\,950\,025}, \frac{47\,976}{49\,950\,025}, \frac{1\,998\,000}{1\,998\,001}\right\}$

ritrova= $\left\{1, \frac{999}{1000}, \frac{1}{3}\right\}$ errore= {0, 0, 0}

Una sperimentazione casuale larghissima

In[103]:=
```
casi = 5000;
Print["Devo fare ", casi, " casi..."];
(* Sperimento le formule stando tra -1 e +
 1 della latitudine e longitudine algebrica *)
j = 0; durata = Timing[
  For[nvolte = 0, casi > nvolte, nvolte++, ss = { RandomInteger[{1, 9}],
     RandomInteger[{-20, 20}] / 10, RandomInteger[{-20, 20}] / 10};
   j++; cc = sn2c[ss]; dd = cn2s[cc] - snor[ss];
   (* Stampa solo nel caso che non ottenga esattamente 0 *)
   If[dd[[1]]^2 + dd[[2]]^2 + dd[[3]]^2 > 0,
    Print[j, ")Errato : ", ss, " errore=", dd, " ...strano..."];]]];
Print["Ha fatto ", j, " casi con durata= ",
  durata[[1]], "\nUltimo caso analizzato= ", ss];
```

```
Devo fare 5000 casi...

Ha fatto 5000 casi con durata= 0.468003
```

Ultimo caso analizzato= $\left\{7, -\frac{4}{5}, \frac{11}{10}\right\}$

Salvo sorprese nella sperimentazione casuale... **tutto funziona bene...**

FUORI TEMA MA INTERESSANTE:
Una serie di versori esprimibili in modo esatto anche in virgola mobile usando 3/5 come seno e 4/5 come coseno iniziale.

In[107]:=
```
s1 = 3 / 5; c1 = 4 / 5; sv = s1; cv = c1; For[j = 1, 12 > j, j++,
 Print[100 + j, ") s= ", N[sv, 14],
  " c= ", N[cv, 14], " Sommaquadrati= ", sv^2 + cv^2];
 sn = sv c1 + cv s1; cn = cv c1 - sv s1;
 sv = sn; cv = cn];
(* Dopo 10 passi ho fatto il giro completo e potrei andare avanti
 ma senza lo scopo di avere numeri esatti anche in virgola mobile *)
```

```
101) s= 0.60000000000000 c= 0.80000000000000 Sommaquadrati= 1
102) s= 0.96000000000000 c= 0.28000000000000 Sommaquadrati= 1
103) s= 0.93600000000000 c= -0.35200000000000 Sommaquadrati= 1
104) s= 0.53760000000000 c= -0.84320000000000 Sommaquadrati= 1
105) s= -0.075840000000000 c= -0.99712000000000 Sommaquadrati= 1
106) s= -0.65894400000000 c= -0.75219200000000 Sommaquadrati= 1
107) s= -0.97847040000000 c= -0.20638720000000 Sommaquadrati= 1
108) s= -0.90660864000000 c= 0.42197248000000 Sommaquadrati= 1
109) s= -0.47210342400000 c= 0.88154316800000 Sommaquadrati= 1
110) s= 0.15124316160000 c= 0.98849658880000 Sommaquadrati= 1
111) s= 0.71409248256000 c= 0.70005137408000 Sommaquadrati= 1
```

Altra formulazione

Questa è la trasformazione di coordinate necessaria per ottenere le coordinate cartesiane.

Notare che è una trasformazione completamente algebrica dove h rappresenta la tangente della mezza latitudine e quindi varia tra 1 e -1 e vale 0 all'equatore ossia a z=0 mentre p rappresenta la tangente del quarto di longitudine e dunque anche lei varia tra 1 e -1 e vale 0 quando y vale 0

In[108]:=

```
dasac[s_] := Block[{r = s[[1]], h = s[[2]], q = s[[3]]},
   {r (1 - h^2) (1 - 6 q^2 + q^4) / ((1 + h^2) (1 + q^2)^2),
    4 q r (1 - h^2) (1 - q^2) / ((1 + h^2) (1 + q^2)^2),
    2 r h / (1 + h^2)}];
dacas[c_] := Block[{x = c[[1]], y = c[[2]], z = c[[3]], r, b, h, q},
   r = Sqrt[x^2 + y^2 + z^2]; If[0 ≥ r , Return[{0, 0, 0}]];
   b = Sqrt[x^2 + y^2]; h = z / (b + r);
   If[0 ≥ b, Return[{r, h, 0}]];
   If[b == -x, Return[{r, h, 1}]];
   q = Sign[y] Sqrt[b - x] / (Sqrt[2 b] + Sqrt[b + x]);
   Simplify[{r, h, q}]];
(* Altra variante che funziona *)
dacasb[c_] := Block[{x = c[[1]], y = c[[2]], z = c[[3]], r, b, h, q, d},
   r = Sqrt[x^2 + y^2 + z^2]; If[0 ≥ r , Return[{0, 0, 0}]];
   b = Sqrt[x^2 + y^2]; h = z / (b + r);
   If[0 ≥ b, Return[{r, h, 0}]];
   d = (x + b + Sqrt[2 b] Sqrt[b + x]);
   If[d == 0, Return[{r, h, 1}]];
   q = y / (x + b + Sqrt[2 b] Sqrt[b + x]);
   Simplify[{r, h, q}]];
sss = {5, 2, 3}; Print[sss];
cc = dasac[sss]; Print[cc]; ss = dacas[cc]; Print[ss]; Print[dasac[ss]];
```

```
{5, 2, 3}
```

$$\left\{-\frac{21}{25}, \frac{72}{25}, 4\right\}$$

$$\left\{5, \frac{1}{2}, \frac{1}{2}\right\}$$

$$\left\{-\frac{21}{25}, \frac{72}{25}, 4\right\}$$

Notare che la funzione dasac[s] produce le coordinate cartesiane giuste anche se s non è stata normalizzata mentre la funzione dacas[c] produce le coordinate algebriche sferiche normalizzate ossia non necessariamente uguali alle coordinate sferiche iniziali se le coordinate sferiche iniziali non erano state preventivamente normalizzate.

Per non complicare le cose ossia dover realizzare una ulteriore funzione di normalizzazione delle coordinate sferiche algebriche penso che la cosa piu' semplice sia trasformare le coordinate sferiche in cartesiane e poi riottenere dalle cartesiane le sferiche ma ovviamente normalizzate dalla doppia trasformazione.

In[113]:=
```
(* Applica questa variante meno efficiente
  d=x+b+Sqrt[2 b]Sqrt[b+x];
If[d==0,Return[{r,h,1}]];
q=y/(x+b+Sqrt[2 b]Sqrt[b+x]);
*)
dacastampa[c_] := Block[{x = c[[1]], y = c[[2]], z = c[[3]], r, b, h, q, d},
   r = Sqrt[x^2 + y^2 + z^2]; If[0 ≥ r , Return[{0, 0, 0}]];
   b = Sqrt[x^2 + y^2]; h = z / (b + r);
   If[0 ≥ b, Return[{r, h, 0}]];
   Print["x =", x, " y=", y, " z= ", z, " r= ", r, " b=", b];
   d = (x + b + Sqrt[2 b] Sqrt[b + x]);
   If[d == 0, Return[{r, h, 1}]];
   Print["(x+b+Sqrt[2 b]Sqrt[b+x]) ", N[(x + b + Sqrt[2 b] Sqrt[b + x])]];
   q = y / (x + b + Sqrt[2 b] Sqrt[b + x]);
   Simplify[{r, h, q}]];
```

Qui sperimenta molto la versione fondamentale:

In[114]:=
```
casi = 1000;
Print["Devo fare ", casi, " casi..."];
(* Sperimento le formule stando tra -1 e +
 1 della latitudine e longitudine algebrica *)
j = 0; durata = Timing[
  For[nvolte = 0, casi > nvolte, nvolte++, ss = { RandomInteger[{1, 9}],
     RandomInteger[{-20, 20}] / 10, RandomInteger[{-20, 20}] / 10};
   j++; (*  Print[j,") ",ss]; *)
   cc = dasac[ss]; ssn = dacas[cc]; ccn = dasac[ssn]; dd = cc - ccn;
   ssnn = dacasb[cc];
   (*
   If[ssn≠ ssnn,Print["\n\n",j,") Attenzione ",cc," fa diversi\n",
      ssn," dal nuovo ",ssnn];Break[];]; *)
   (* Stampa solo nel caso che non ottenga esattamente 0 *)
   vedo = 0; If[5 > j, vedo = -1];
   If[dd[[1]]^2 + dd[[2]]^2 + dd[[3]]^2 > vedo,
    Print[j, ")ss= ", ss, " ssn= ", ssn,
      "\ncc =", cc, " ccn= ", ccn, "  errore= ", dd, "..."];];
   If[ssn ≠ ssnn, Print["\n\n", j, ") Attenzione ", cc, " fa diversi\n",
     ssn, " dal nuovo ", ssnn]; Break[];];
  ]];
Print["Ha fatto ", j, " casi con durata= ",
  durata[[1]], "\nUltimo caso analizzato= ", ss];
```

```
Devo fare 1000 casi...
```

1) ss= $\left\{8, -\frac{17}{10}, \frac{1}{10}\right\}$ ssn= $\left\{8, -\frac{10}{17}, -\frac{9}{11}\right\}$

cc = $\left\{-\frac{14\,214\,312}{3\,968\,189}, -\frac{5\,987\,520}{3\,968\,189}, -\frac{2720}{389}\right\}$ ccn= $\left\{-\frac{14\,214\,312}{3\,968\,189}, -\frac{5\,987\,520}{3\,968\,189}, -\frac{2720}{389}\right\}$ errore= {0, 0, 0}...

2) ss= $\left\{1, -\frac{4}{5}, -\frac{6}{5}\right\}$ ssn= $\left\{1, -\frac{4}{5}, \frac{5}{6}\right\}$

cc = $\left\{-\frac{31\,311}{152\,561}, \frac{11\,880}{152\,561}, -\frac{40}{41}\right\}$ ccn= $\left\{-\frac{31\,311}{152\,561}, \frac{11\,880}{152\,561}, -\frac{40}{41}\right\}$ errore= {0, 0, 0}...

3) ss= $\left\{6, -\frac{3}{10}, \frac{9}{10}\right\}$ ssn= $\left\{6, -\frac{3}{10}, \frac{9}{10}\right\}$

cc = $\left\{-\frac{17\,493\,294}{3\,570\,949}, \frac{3\,734\,640}{3\,570\,949}, -\frac{360}{109}\right\}$ ccn= $\left\{-\frac{17\,493\,294}{3\,570\,949}, \frac{3\,734\,640}{3\,570\,949}, -\frac{360}{109}\right\}$ errore= {0, 0, 0}...

4) ss= $\left\{3, -\frac{8}{5}, \frac{4}{5}\right\}$ ssn= $\left\{3, -\frac{5}{8}, -\frac{1}{9}\right\}$

cc = $\left\{\frac{177\,723}{149\,609}, -\frac{84\,240}{149\,609}, -\frac{240}{89}\right\}$ ccn= $\left\{\frac{177\,723}{149\,609}, -\frac{84\,240}{149\,609}, -\frac{240}{89}\right\}$ errore= {0, 0, 0}...

```
Ha fatto 1000 casi con durata= 0.421203
```

Ultimo caso analizzato= $\left\{7, -\frac{3}{2}, -\frac{17}{10}\right\}$

DOPO UNA FATICOSA RICERCA PER TROVARE LE FORMULE PIU' SEMPLICI POSSIBILI

Ecco una solidissima formula per calcolare h° e q° date le coordinate cartesiane x°, y° e z°.

In[118]:=
```
h° = z° / (Sqrt[x°^2 + y°^2 + z°^2] + Sqrt[x°^2 + y°^2]);
Print["h° = ", h°];
q° = FullSimplify[y° / (x° + Sqrt[x°^2 + y°^2] +
       Sqrt[2 Sqrt[x°^2 + y°^2]] Sqrt[Sqrt[x°^2 + y°^2] + x°] )];
Print["q° = ", q°];
```

$$h^\circ = \frac{z^\circ}{\sqrt{x^{\circ 2}+y^{\circ 2}}+\sqrt{x^{\circ 2}+y^{\circ 2}+z^{\circ 2}}}$$

$$q^\circ = \frac{y^\circ}{x^\circ+\sqrt{x^{\circ 2}+y^{\circ 2}}+\sqrt{2}\,\left(x^{\circ 2}+y^{\circ 2}\right)^{1/4}\sqrt{x^\circ+\sqrt{x^{\circ 2}+y^{\circ 2}}}}$$

In[122]:=
```
aaa = FullSimplify[r (1 - h^2) (1 - 6 q^2 + q^4) / ((1 + h^2) (1 + q^2)^2) /.
      r → Sqrt[x°^2 + y°^2 + z°^2] /. h → h°];
FullSimplify[aaa /. q → q°]
```

Out[123]= x°

In[124]:=
```
bbb =
 FullSimplify[(4 * (1 - h^2) * q * (1 - q^2) * r) / ((1 + h^2) * (1 + q^2)^2) /.
   r -> Sqrt[x°^2 + y°^2 + z°^2] /. h -> h°]; FullSimplify[bbb /. q → q°]
```

Out[124]= y°

Notare che il calcolo di q° crea problemi numerici solo se x° ed y° sono entrambi nulli ma questa è una situazione veramente eccezionale perché in quel caso h° vale 1 se z° è positivo e vale -1 se z° è negativo. Si può decidere di far valere h°=0 se anche z° è nullo come le altre due coordinate cartesiane.

Altro modo di calcolare q. Questo, q°° mi sembra più solido del modo q°.

In[125]:=
```
q°° = Sign[y°] Sqrt[Sqrt[x°^2 + y°^2] - x°] /
    ( Sqrt[Sqrt[x°^2 + y°^2] + x°] + Sqrt[2 Sqrt[x°^2 + y°^2]] )
```

Out[125]=
$$\frac{\sqrt{-x^\circ+\sqrt{x^{\circ 2}+y^{\circ 2}}}\;\text{Sign}[y^\circ]}{\sqrt{2}\,\left(x^{\circ 2}+y^{\circ 2}\right)^{1/4}+\sqrt{x^\circ+\sqrt{x^{\circ 2}+y^{\circ 2}}}}$$

In[126]:= `q°`

Out[126]=
$$\frac{y^\circ}{x^\circ+\sqrt{x^{\circ 2}+y^{\circ 2}}+\sqrt{2}\,\left(x^{\circ 2}+y^{\circ 2}\right)^{1/4}\sqrt{x^\circ+\sqrt{x^{\circ 2}+y^{\circ 2}}}}$$

Una verifica numerica che i due metodi sono identici ... usando molte cifre significative...

In[127]:= `N[(q°°), 30] /. x° → -15 /. y° → 0`

Out[127]= 0

In[128]:= `? dasac`

```
Global`dasac
```

```
dasac[s_] := Block[{r = s[[1]], h = s[[2]], q = s[[3]]}, {r (1-h^2) (1-6 q^2+q^4) / ((1+h^2) (1+q^2)^2), 4 q r (1-h^2) (1-q^2) / ((1+h^2) (1+q^2)^2), 2 r h / (1+h^2)}]
```

In[129]:=
```
? dacas
```

```
Global`dacas
```

```
dacas[c_] := Block[{x = c[[1]], y = c[[2]], z = c[[3]], r, b, h, q}, r = Sqrt[x^2 + y^2 + z^2];
  If[0 ≥ r, Return[{0, 0, 0}]]; b = Sqrt[x^2 + y^2]; h = z / (b+r); If[0 ≥ b, Return[{r, h, 0}]];
  If[b == -x, Return[{r, h, 1}]]; q = Sign[y] Sqrt[b-x] / (Sqrt[2 b] + Sqrt[b+x]); Simplify[{r, h, q}]]
```

Trasformazioni di coordinate.

Uso qui, come metriche di test, i casi illustrati precedentemente.

In questo caso la matrice di trasformazione NON è simmetrica come si vedrà in questi esempi ossia nella trasformazione da coordinate cartesiane a coordinate sferiche:

In[130]:=
```
coord = {r, h, q}; dcoord = {dr, dh, dq};
isc = {r (1 - h^2) (1 - 6 q^2 + q^4) / ((1 + h^2) (1 + q^2)^2),
   4 q r (1 - h^2) (1 - q^2) / ((1 + h^2) (1 + q^2)^2), 2 r h / (1 + h^2)}
```

Out[131]=
$$\left\{\frac{\left(1-h^2\right)\left(1-6\,q^2+q^4\right)r}{\left(1+h^2\right)\left(1+q^2\right)^2},\ \frac{4\left(1-h^2\right)q\left(1-q^2\right)r}{\left(1+h^2\right)\left(1+q^2\right)^2},\ \frac{2\,h\,r}{1+h^2}\right\}$$

Calcola la matrice iacobiana di questa trasformazione di coordinate.

In[132]:=
```
iacob = Simplify[D[isc, {coord}]]; MatrixForm[iacob]
```

Out[132]//MatrixForm=
$$\begin{pmatrix} -\frac{\left(-1+h^2\right)\left(1-6\,q^2+q^4\right)}{\left(1+h^2\right)\left(1+q^2\right)^2} & -\frac{4\,h\left(1-6\,q^2+q^4\right)r}{\left(1+h^2\right)^2\left(1+q^2\right)^2} & -\frac{16\left(-1+h^2\right)q\left(-1+q^2\right)r}{\left(1+h^2\right)\left(1+q^2\right)^3} \\ \frac{4\left(-1+h^2\right)q\left(-1+q^2\right)}{\left(1+h^2\right)\left(1+q^2\right)^2} & \frac{16\,h\,q\left(-1+q^2\right)r}{\left(1+h^2\right)^2\left(1+q^2\right)^2} & -\frac{4\left(-1+h^2\right)\left(1-6\,q^2+q^4\right)r}{\left(1+h^2\right)\left(1+q^2\right)^3} \\ \frac{2\,h}{1+h^2} & -\frac{2\left(-1+h^2\right)r}{\left(1+h^2\right)^2} & 0 \end{pmatrix}$$

Ecco il tensore metrico che viene fuori dallo jacobiano. Notare che è una matrice diagonale ossia r, h e q sono coordinate tra loro ortogonali. Come si è visto 1>=Abs[h] ed anche 1 >=Abs[q] ma è ammesso usare coordinate non normalizzate ossia h e q maggiori, in valore assoluto, di 1.

In[133]:=
```
gdd = Simplify[Transpose[iacob].iacob]; Print[MatrixForm[gdd]];
```

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{4\,r^2}{\left(1+h^2\right)^2} & 0 \\ 0 & 0 & \frac{16\left(-1+h^2\right)^2 r^2}{\left(1+h^2\right)^2\left(1+q^2\right)^2} \end{pmatrix}$$

Trovo notevole il fatto che l'intera matrice gdd sia data da funzioni razionali... Il determinante è ovunque definito e positivo salvo nel punto Abs[h]==1 ossia l'asse polare del sistema di coordinate sferiche.

In[134]:=
```
Simplify[Det[gdd]]
```

Out[134]=
$$\frac{64\left(-1+h^2\right)^2 r^4}{\left(1+h^2\right)^4\left(1+q^2\right)^2}$$

In[135]:=
```
trhq = {r, 2 ArcTan[h], 4 ArcTan[q]}
```

Out[135]=
```
{r, 2 ArcTan[h], 4 ArcTan[q]}
```

In[136]:=
```
iacob°trhq = D[trhq, {coord}]; Print[MatrixForm[iacob°trhq]];
```

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{2}{1+h^2} & 0 \\ 0 & 0 & \frac{4}{1+q^2} \end{pmatrix}$$

In[137]:=
```
gs2 = DiagonalMatrix[FullSimplify[TrigExpand[{1, rr^2, (rr Cos[hh])^2}]]];
Print[MatrixForm[gs2]];
```

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & rr^2 & 0 \\ 0 & 0 & rr^2\,\mathrm{Cos}[hh]^2 \end{pmatrix}$$

In[138]:=
```
Transpose[iacob°trhq].gs2.iacob°trhq /. rr → r
```

Out[138]=
$$\left\{\{1, 0, 0\}, \left\{0, \frac{4\,r^2}{\left(1+h^2\right)^2}, 0\right\}, \left\{0, 0, \frac{16\,r^2\,\mathrm{Cos}[hh]^2}{\left(1+q^2\right)^2}\right\}\right\}$$

In[139]:=
```
ddx = iacob.dcoord;
```

Per verifica...

In[140]:=
```
Simplify[ddx.ddx - dcoord.gdd.dcoord]
```

Out[140]=
```
0
```

In[141]:=
```
Simplify[isc.isc]
```

Out[141]=
$$r^2$$

In[142]:=
```
Simplify[isc[[1 ;; 2]].isc[[1 ;; 2]]]
```

Out[142]=
$$\frac{\left(-1+h^2\right)^2 r^2}{\left(1+h^2\right)^2}$$

In[143]:=
```
xx =
 r (1 - h^2) (1 - 6 qq + qq^2) / ((1 + h^2) (1 + qq)^2) /. r → Sqrt[x^2 + y^2 + z^2] /.
  h → (Sqrt[x^2 + y^2 + z^2] - Sqrt[x^2 + y^2]) / z; Simplify[xx]
```

Out[143]=
$$\frac{\left(1-6\,qq+qq^2\right)\sqrt{x^2+y^2+z^2}\left(1-\frac{\left(\sqrt{x^2+y^2}-\sqrt{x^2+y^2+z^2}\right)^2}{z^2}\right)}{(1+qq)^2\left(1+\frac{\left(\sqrt{x^2+y^2}-\sqrt{x^2+y^2+z^2}\right)^2}{z^2}\right)}$$

In[144]:=
```
Solve[(1 - 6 qq + qq^2) ope / (1 + qq)^2 - x == 0, qq]
```

Out[144]=
$$\left\{\left\{qq \to \frac{3\,ope + x - 2\sqrt{2}\,\sqrt{ope^2 + ope\,x}}{ope - x}\right\}, \left\{qq \to \frac{3\,ope + x + 2\sqrt{2}\,\sqrt{ope^2 + ope\,x}}{ope - x}\right\}\right\}$$

In[145]:=
```
Solve[xx - x == 0, qq]
```

Out[145]=
$$\left\{\left\{qq \to \left(-2x^3-2xy^2-6x^2\sqrt{x^2+y^2}-6y^2\sqrt{x^2+y^2}-2xz^2-6\sqrt{x^2+y^2}\,z^2+6x^2\sqrt{x^2+y^2+z^2}+6y^2\sqrt{x^2+y^2+z^2}+2x\sqrt{x^2+y^2}\,\sqrt{x^2+y^2+z^2}-\sqrt{\left(\left(2x^3+2xy^2+6x^2\sqrt{x^2+y^2}+6y^2\sqrt{x^2+y^2}+2xz^2+6\sqrt{x^2+y^2}\,z^2-6x^2\sqrt{x^2+y^2+z^2}-6y^2\sqrt{x^2+y^2+z^2}-2x\sqrt{x^2+y^2}\,\sqrt{x^2+y^2+z^2}\right)^2-4\left(x^3+xy^2-x^2\sqrt{x^2+y^2}-y^2\sqrt{x^2+y^2}+xz^2-\sqrt{x^2+y^2}\,z^2+x^2\sqrt{x^2+y^2+z^2}+y^2\sqrt{x^2+y^2+z^2}-x\sqrt{x^2+y^2}\,\sqrt{x^2+y^2+z^2}\right)^2\right)\right)\Big/\left(2\left(x^3+xy^2-x^2\sqrt{x^2+y^2}-y^2\sqrt{x^2+y^2}+xz^2-\sqrt{x^2+y^2}\,z^2+x^2\sqrt{x^2+y^2+z^2}+y^2\sqrt{x^2+y^2+z^2}-x\sqrt{x^2+y^2}\,\sqrt{x^2+y^2+z^2}\right)\right)\right\},$$
$$\left\{qq \to \left(-2x^3-2xy^2-6x^2\sqrt{x^2+y^2}-6y^2\sqrt{x^2+y^2}-2xz^2-6\sqrt{x^2+y^2}\,z^2+6x^2\sqrt{x^2+y^2+z^2}+6y^2\sqrt{x^2+y^2+z^2}+2x\sqrt{x^2+y^2}\,\sqrt{x^2+y^2+z^2}+\sqrt{\left(\left(2x^3+2xy^2+6x^2\sqrt{x^2+y^2}+6y^2\sqrt{x^2+y^2}+2xz^2+6\sqrt{x^2+y^2}\,z^2-6x^2\sqrt{x^2+y^2+z^2}-6y^2\sqrt{x^2+y^2+z^2}-2x\sqrt{x^2+y^2}\,\sqrt{x^2+y^2+z^2}\right)^2-4\left(x^3+xy^2-x^2\sqrt{x^2+y^2}-y^2\sqrt{x^2+y^2}+xz^2-\sqrt{x^2+y^2}\,z^2+x^2\sqrt{x^2+y^2+z^2}+y^2\sqrt{x^2+y^2+z^2}-x\sqrt{x^2+y^2}\,\sqrt{x^2+y^2+z^2}\right)^2\right)\right)\Big/\left(2\left(x^3+xy^2-x^2\sqrt{x^2+y^2}-y^2\sqrt{x^2+y^2}+xz^2-\sqrt{x^2+y^2}\,z^2+x^2\sqrt{x^2+y^2+z^2}+y^2\sqrt{x^2+y^2+z^2}-x\sqrt{x^2+y^2}\,\sqrt{x^2+y^2+z^2}\right)\right)\right\}\right\}$$

Da sferiche in cartesiane senza particolari accorgimenti

L'unica avvertenza è quella di trattare in modo speciale il caso in cui sia x che y sono nulli.
In tal caso si attribuisce ad x un valore infinitesimo e si procede poi normalmente ossia, in forma generale, se $0 \geq$ Abs[x] allora modifico l'ascissa di una quantità infinitesima ossia x=1/10^100 ossia non uso mai lo 0 ma assegno ad x un incremento irrilevante dal punto di vista numerico... e tutto il resto segue di conseguenza senza pericolo di divisioni per 0.

In[146]:=
```
(* Definisco i simboli delle coordinate cartesiane *)
carte = {x, y, z};
(* Definisco le funzioni di trasformazione da cartesiane in
 sferiche applicabili a patto che x non sia esattamente zero,
altrimenti uso un x positivo ma veramente trascurabile *)
insferico = {Sqrt[x^2 + y^2 + z^2],
   ArcSin[z / Sqrt[x^2 + y^2 + z^2]], ArcSin[y / Sqrt[x^2 + y^2]]};
Print["Trasformazione classica da cartesiano in sferico\n", insferico];
jacosfer = FullSimplify[D[insferico, {carte}]];
Print["Matrice Jacobiana desunta:\n", MatrixForm[jacosfer]];
```

Trasformazione classica da cartesiano in sferico

$$\left\{\sqrt{x^2+y^2+z^2}\,,\ \text{ArcSin}\left[\frac{z}{\sqrt{x^2+y^2+z^2}}\right],\ \text{ArcSin}\left[\frac{y}{\sqrt{x^2+y^2}}\right]\right\}$$

Matrice Jacobiana desunta:

$$\begin{pmatrix} \frac{x}{\sqrt{x^2+y^2+z^2}} & \frac{y}{\sqrt{x^2+y^2+z^2}} & \frac{z}{\sqrt{x^2+y^2+z^2}} \\ -\frac{x\,z}{\sqrt{\frac{x^2+y^2}{x^2+y^2+z^2}}\,\left(x^2+y^2+z^2\right)^{3/2}} & -\frac{y\,z}{\sqrt{\frac{x^2+y^2}{x^2+y^2+z^2}}\,\left(x^2+y^2+z^2\right)^{3/2}} & \frac{\sqrt{\frac{x^2+y^2}{x^2+y^2+z^2}}}{\sqrt{x^2+y^2+z^2}} \\ -\frac{x\,y}{\sqrt{\frac{x^2}{x^2+y^2}}\,\left(x^2+y^2\right)^{3/2}} & \frac{\sqrt{\frac{x^2}{x^2+y^2}}}{\sqrt{x^2+y^2}} & 0 \end{pmatrix}$$

La matrice jacobiana DUNQUE NON è evidentemente simmetrica.

La trasposta della matrice jacobiana fatta una volta per tutte...

In[150]:=
```
traspojacosfer = Transpose[jacosfer]; Print[MatrixForm[traspojacosfer]];
```

$$\begin{pmatrix} \frac{x}{\sqrt{x^2+y^2+z^2}} & -\frac{x\,z}{\sqrt{\frac{x^2+y^2}{x^2+y^2+z^2}}\,\left(x^2+y^2+z^2\right)^{3/2}} & -\frac{x\,y}{\sqrt{\frac{x^2}{x^2+y^2}}\,\left(x^2+y^2\right)^{3/2}} \\ \frac{y}{\sqrt{x^2+y^2+z^2}} & -\frac{y\,z}{\sqrt{\frac{x^2+y^2}{x^2+y^2+z^2}}\,\left(x^2+y^2+z^2\right)^{3/2}} & \frac{\sqrt{\frac{x^2}{x^2+y^2}}}{\sqrt{x^2+y^2}} \\ \frac{z}{\sqrt{x^2+y^2+z^2}} & \frac{\sqrt{\frac{x^2+y^2}{x^2+y^2+z^2}}}{\sqrt{x^2+y^2+z^2}} & 0 \end{pmatrix}$$

Ma io so, da altre fonti che questo è il tensore metrico diagonale classico...

In[151]:=
```
madiago2 = {1, ρ^2, (ρ Cos[θ])^2};
gsfer2 = DiagonalMatrix[madiago2]; Print[MatrixForm[gsfer2]]
```

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \rho^2 & 0 \\ 0 & 0 & \rho^2\,\text{Cos}[\theta]^2 \end{pmatrix}$$

Ora dimostro che la definizione di madiago2 è giusta ossia partendo da gsfer2 ossia il tensore metrico sferico, riottengo il tensore metrico cartesiano ossia la matrice identità di ordine 3.

Come primo passo esprimo la matrice in funzione delle coordinate cartesiane ottenendo però una matrice ibrida ...

In[152]:=
```
primopasso = FullSimplify[
    gsfer2 /. ρ → insferico[[1]] /. θ → insferico[[2]] /. ϕ → insferico[[3]]];
MatrixForm[primopasso]
```

Out[153]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & x^2+y^2+z^2 & 0 \\ 0 & 0 & x^2+y^2 \end{pmatrix}$$

Poi finalmente applico la matrice jacobiana e la sua trasposta ed ottengo quello che mi aspetto...

In[154]:=
```
Simplify[traspojacosfer.primopasso.jacosfer]
```

Out[154]=
```
{{1, 0, 0}, {0, 1, 0}, {0, 0, 1}}
```

Circa il come cambiare vettori e tensori cambiando il sistema di riferimento...

Questa espressione la devo usare come schema per ottenere in generale un qualsiasi tensore metrico in un nuovo sistema di coordinate.

1. Sostituisco nel tensore che ho, alle coordinate vecchie le coordinate nuove usando le funzioni che esprimono le coordinate vecchie come funzioni delle coordinate nuove.
2. Calcolo lo jacobiano ossia derivo le funzioni che esprimono le coordinate vecchie in funzione delle coordinate nuove rispetto alle varie coordinate nuove. Ottengo quindi una matrice di solito NON simmetrica. Ogni riga contiene una derivata parziale di una stessa funzione ed ogni colonna contiene la derivata parziale delle varie funzioni rispetto ad una stessa coordinata nuova.
3. **Il nuovo vettore covariante si ottiene moltiplicando la trasposta della matrice Jacobiana per il vecchio vettore covariante**.
4. **Il nuovo vettore controvariante si ottiene moltiplicando la matrice inversa della matrice Jacobiana per il vecchio vettore controvariante**.

In[155]:=
```
jacobiano = {{j11, j12, j13}, {j21, j22, j23}, {j31, j32, j33}};
MatrixForm[jacobiano]
```

Out[155]//MatrixForm=

$$\begin{pmatrix} j11 & j12 & j13 \\ j21 & j22 & j23 \\ j31 & j32 & j33 \end{pmatrix}$$

In[156]:=
```
gv = {{gv11, gv12, gv13}, {gv21, gv22, gv23}, {gv31, gv32, gv33}};
gvecchio2 = (gv + Transpose[gv]) / 2;
dv11 = {dv1, dv2, dv3};
Print[MatrixForm[gvecchio2]];
Print[dv11];
```

$$\begin{pmatrix} gv11 & \frac{gv12+gv21}{2} & \frac{gv13+gv31}{2} \\ \frac{gv12+gv21}{2} & gv22 & \frac{gv23+gv32}{2} \\ \frac{gv13+gv31}{2} & \frac{gv23+gv32}{2} & gv33 \end{pmatrix}$$

```
{dv1, dv2, dv3}
```

In[160]:=
```
altrog2 = Transpose[jacobiano].gvecchio2.jacobiano;
Print[MatrixForm[Simplify[altrog2 - Transpose[altrog2]]]];
```

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Il cambiamento di coordinate per un vettore controvariante si fa prendendo l'inversa dello jacobiano per il vecchio vettore controvariante.

In[162]:=
```
altrov11 = Simplify[Inverse[jacobiano].dv11]
```

Out[162]=
$$\left\{\frac{dv3\ j13\ j22 - dv3\ j12\ j23 - dv2\ j13\ j32 + dv1\ j23\ j32 + dv2\ j12\ j33 - dv1\ j22\ j33}{j13\ j22\ j31 - j12\ j23\ j31 - j13\ j21\ j32 + j11\ j23\ j32 + j12\ j21\ j33 - j11\ j22\ j33},\right.$$
$$\frac{dv3\ j13\ j21 - dv3\ j11\ j23 - dv2\ j13\ j31 + dv1\ j23\ j31 + dv2\ j11\ j33 - dv1\ j21\ j33}{-j13\ j22\ j31 + j12\ j23\ j31 + j13\ j21\ j32 - j11\ j23\ j32 - j12\ j21\ j33 + j11\ j22\ j33},$$
$$\left.\frac{dv3\ j12\ j21 - dv3\ j11\ j22 - dv2\ j12\ j31 + dv1\ j22\ j31 + dv2\ j11\ j32 - dv1\ j21\ j32}{j13\ j22\ j31 - j12\ j23\ j31 - j13\ j21\ j32 + j11\ j23\ j32 + j12\ j21\ j33 - j11\ j22\ j33}\right\}$$

ATTENZIONE: **PER FARE IL NUOVO VETTORE CONTROVARIANTE VA PRESA L'INVERSA DELLA MATRICE JACOBIANA !**

Questo prodotto tra vettore controvariante e tensore metrico covariante e vettore controvariante DEVE essere un invariante ossia una quantità scalare che NON DEVE DIPENDERE dallo jacobiano.

In[163]:=
```
scal = Simplify[dv11.gvecchio2.dv11]
```

Out[163]=
$$dv1^2\ gv11 + dv2^2\ gv22 + dv1\ (dv2\ (gv12 + gv21) + dv3\ (gv13 + gv31)) + dv2\ dv3\ (gv23 + gv32) + dv3^2\ gv33$$

Il prodotto del tensore metrico covariante per un vettore controvariante dà un vettore covariante.

In[164]:=
```
dv12 = gvecchio2.dv11;
```

Il cambiamento di coordinate per un vettore covariante si fa prendendo la trasposta dello jacobiano per il vecchio vettore covariante.
Lo faccio e dimostro che questa affermazione è corretta.

In[165]:=
```
altrov12 = Simplify[Transpose[jacobiano].dv12]
```

Out[165]=
$$\left\{\frac{1}{2}\ ((2\ dv1\ gv11 + dv2\ (gv12 + gv21) + dv3\ (gv13 + gv31))\ j11 + (dv1\ (gv12 + gv21) + 2\ dv2\ gv22 + dv3\ (gv23 + gv32))\ j21 + (dv1\ (gv13 + gv31) + dv2\ (gv23 + gv32) + 2\ dv3\ gv33)\ j31),\right.$$
$$\frac{1}{2}\ ((2\ dv1\ gv11 + dv2\ (gv12 + gv21) + dv3\ (gv13 + gv31))\ j12 + (dv1\ (gv12 + gv21) + 2\ dv2\ gv22 + dv3\ (gv23 + gv32))\ j22 + (dv1\ (gv13 + gv31) + dv2\ (gv23 + gv32) + 2\ dv3\ gv33)\ j32),$$
$$\left.\frac{1}{2}\ ((2\ dv1\ gv11 + dv2\ (gv12 + gv21) + dv3\ (gv13 + gv31))\ j13 + (dv1\ (gv12 + gv21) + 2\ dv2\ gv22 + dv3\ (gv23 + gv32))\ j23 + (dv1\ (gv13 + gv31) + dv2\ (gv23 + gv32) + 2\ dv3\ gv33)\ j33)\right\}$$

ATTENZIONE: **PER FARE IL NUOVO VETTORE COVARIANTE VA PRESA LA TRASPOSTA DELLA MATRICE JACOBIANA** E NON quella calcolata !

Il prodotto scalare tra il nuovo vettore controvariante e il nuovo vettore covariante deve essere uguale al trodotto scalare tra il vecchio vettore controvariante ed il vecchio vettore covariante.

In[166]:=
```
Simplify[altrov11.altrov12]
```

Out[166]=
$$dv1^2\, gv11 + dv2^2\, gv22 + dv1\,(dv2\,(gv12 + gv21) + dv3\,(gv13 + gv31)) + dv2\, dv3\,(gv23 + gv32) + dv3^2\, gv33$$

Scompongo tutti i fattori del calcolo:

In[167]:=
```
Simplify[dv11.Transpose[Inverse[jacobiano]].
  (Transpose[jacobiano].gvecchio2.jacobiano).Inverse[jacobiano].dv11]
```

Out[167]=
$$dv1^2\, gv11 + dv2^2\, gv22 + dv1\,(dv2\,(gv12 + gv21) + dv3\,(gv13 + gv31)) + dv2\, dv3\,(gv23 + gv32) + dv3^2\, gv33$$

Raccolgo qualche fattore...

In[168]:=
```
Simplify[altrov11.(Transpose[jacobiano].gvecchio2.jacobiano).altrov11]
```

Out[168]=
$$dv1^2\, gv11 + dv2^2\, gv22 + dv1\,(dv2\,(gv12 + gv21) + dv3\,(gv13 + gv31)) + dv2\, dv3\,(gv23 + gv32) + dv3^2\, gv33$$

Ed evidenzio tutti i vettori nuovi ossia il nuovo vettore controvariante, il nuovo tensore metrico covariante ed il nuovo vettore controvariante.
Devo riottenere ancora lo stesso scalare !

In[169]:=
```
altroscal = Simplify[altrov11.altrog2.altrov11]
```

Out[169]=
$$dv1^2\, gv11 + dv2^2\, gv22 + dv1\,(dv2\,(gv12 + gv21) + dv3\,(gv13 + gv31)) + dv2\, dv3\,(gv23 + gv32) + dv3^2\, gv33$$

Usando numeri casuali ecco qualche verifica - ESPERIMENTO....

In[170]:=
```
njacobiana =
  RandomInteger[{-9, 99}, {4, 4}] + DiagonalMatrix[{500, -500, -500, -500}];
Print["La matrice jacobiana ", MatrixForm[njacobiana]];
ngv = RandomInteger[{-9, 99}, {4, 4}] + DiagonalMatrix[{500, -500, -500, -500}];
ngvecchio2 = (ngv + Transpose[ngv]) / 2;
ndv11 = RandomInteger[{-9, 99}, {4}];
Print["Il vecchio tensore metrico covariante ", MatrixForm[ngvecchio2]];
Print["Un vettore controvariante a caso... vecchio ", ndv11];
naltrog2 = Transpose[njacobiana].ngvecchio2.njacobiana;
Print["Il nuovo tensore metrico covariante ",
  MatrixForm[Simplify[naltrog2]]];
naltrov11 = Simplify[Inverse[njacobiana].ndv11];
nscal = Simplify[ndv11.ngvecchio2.ndv11];
Print["Questo sarebbe il prodotto scalare tra i
    due vettori nel sistema di coordinate VECCHIO: ", nscal];
Print["Il vettore controvariane nel nuovo sistema di riferimento ",
  naltrov11];
naltroscal = Simplify[naltrov11.naltrog2.naltrov11];
Print["Questo sarebbe il prodotto scalare tra i due
    vettori nel sistema di coordinate NUOVO: ", naltroscal];
(* ------- *)
```

La matrice jacobiana $\begin{pmatrix} 584 & 51 & 81 & 22 \\ 67 & -497 & 50 & 28 \\ -7 & 83 & -486 & 55 \\ 16 & 78 & -6 & -449 \end{pmatrix}$

Il vecchio tensore metrico covariante $\begin{pmatrix} 494 & \frac{35}{2} & 56 & 37 \\ \frac{35}{2} & -459 & \frac{163}{2} & 24 \\ 56 & \frac{163}{2} & -437 & \frac{101}{2} \\ 37 & 24 & \frac{101}{2} & -414 \end{pmatrix}$

Un vettore controvariante a caso... vecchio {55, 53, 80, 16}

Il nuovo tensore metrico covariante $\begin{pmatrix} 167\,860\,593 & \frac{59\,672\,205}{2} & \frac{3\,837\,267}{2} & 817\,607 \\ \frac{59\,672\,205}{2} & -125\,670\,990 & \frac{96\,137\,857}{2} & 20\,459\,478 \\ \frac{3\,837\,267}{2} & \frac{96\,137\,857}{2} & -109\,122\,912 & 18\,737\,916 \\ 817\,607 & 20\,459\,478 & 18\,737\,916 & -88\,326\,022 \end{pmatrix}$

Questo sarebbe il prodotto scalare tra
i due vettori nel sistema di coordinate VECCHIO: -1 176 716

Il vettore controvariane nel nuovo sistema di riferimento
$\left\{\frac{8\,274\,509\,319}{62\,636\,165\,926}, -\frac{6\,934\,393\,939}{62\,636\,165\,926}, -\frac{11\,951\,409\,005}{62\,636\,165\,926}, -\frac{1\,491\,047\,638}{31\,318\,082\,963}\right\}$

Questo sarebbe il prodotto scalare tra i due vettori nel sistema di coordinate NUOVO:
-1 176 716

Conclusione 1

Questa è una conclusione su modi alternativi di definire le coordinate sferiche.

Ritengo particolarmente interessante questa funzione ossia la trasformazione tra coordinate algebriche e coordinate cartesiane:

In[184]:= **? sn2c**

```
Global`sn2c
```

```
sn2c[s_] := Block[{th, tq, sm, cm, snp, cnp, snh, cnh}, th = s[[2]]^2; snh = 2 s[[2]]/(1+th); cnh = (1-th)/(1+th);
  tq = s[[3]]^2; sm = 2 s[[3]]/(1+tq); cm = (1-tq)/(1+tq); snp = 2 sm cm; cnp = cm^2 - sm^2; s[[1]] {cnh cnp, cnh snp, snh}]
```

La trasformazione inversa è

In[185]:= **? cn2s**

```
Global`cn2s
```

```
cn2s[c_] := Block[{x = c[[1]], y = c[[2]], z = c[[3]], rq, rr, rz, rb, tp = 0, th = 0},
  rq = x^2 + y^2; rz = Sqrt[rq + z^2]; If[rz > 0, rr = Sqrt[rq]; th = z/(rr+rz); rb = Sqrt[2 rq + 2 Abs[x] rr];
   If[rb > 0, If[x > 0, tp = y/(x+rr+rb), If[y == 0, tp = 1, tp = Sign[y] (Abs[x]+rr)/(Abs[y]+rb)]]]]; {rz, th, tp}]
```

Per normalizzare le coordinate della rappresentazione sferica è data da:

In[186]:= **? snor**

```
Global`snor
```

```
snor[s_] := Block[{th = s[[2]], tt = s[[3]]},
  If[Abs[th] == 1, tt = 0]; If[th > 1, th = 1/th; If[tt == -1, tt = 0, tt = (tt-1)/(1+tt)]];
  If[-1 > th, th = 1/th; If[tt == -1, tt = 0, tt = (tt-1)/(1+tt)]];
  If[-1 ≥ tt, tt = -1/tt]; If[tt > 1, tt = -1/tt]; {s[[1]], th, tt}]
```

Il tensore metrico che ne consegue, è una matrice diagonale di funzioni razionali:

In[187]:=
```
Print[MatrixForm[gdd]]
```

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{4\,r^2}{\left(1+h^2\right)^2} & 0 \\ 0 & 0 & \frac{16\left(-1+h^2\right)^2 r^2}{\left(1+h^2\right)^2\left(1+q^2\right)^2} \end{pmatrix}$$

In[188]:=
```
hh = 2 ArcTan[h]; qq = 4 ArcTan[q]
```

Out[188]=
```
4 ArcTan[q]
```

In[189]:=
```
D[hh, h]
```

Out[189]=
$$\frac{2}{1+h^2}$$

In[190]:=
```
D[qq, q]
```

Out[190]=
$$\frac{4}{1+q^2}$$

Conclusione

Molto più semplice e veloce fare così (il cambiamento di coordinate senza particolari accorgimenti) anche se faccio uso di funzioni trigonometriche quali l' ArcSin[ ] e il Cos[ ]

IMPORTANTE DA RICORDARE LA REGOLA GENERALE DEL CAMBIAMENTO DELLE COORDINATE...

Sia **Jnv la matrice Jacobiana** ottenuta derivando le funzioni delle vecchie coordinate espresse in funzione delle nuove.

Se Vv1 è un vettore covariante vecchio e Vv11 lo stesso vettore ma in forma controvariante allora ottengo il nuovo vettore Vn1 covariante e il nuovo vettore Vn11 ma in forma controvariante con queste trasformazioni:

```
Vn1 = Dot[ Transpose[ Jnv ], Vv1];
Vn11 = Dot[ Inverse[ Jnv ], Vv11];
```

Ossia usare la trasposta della matrice Jacobiana nel cambiamento di un vettore covariante e l'inversa nel cambiamento di un vettore ocvariante.

Da queste formule appare subito evidente che :

```
Dot[ Vv1, Vv11] = Dot[ Vv11, Vv1] = Dot[ Vn1, Vn11] = Dot[ Vn11, Vn1]
```

L'ovvia estensione della trasformazione per tensori di rango 2 (tra cui è importantissimo il tensore metrico ma il tensore metrico è uno solo dei tanti tensori di rango due concepibili), si fa con queste formule detto Tv2 un tensore di rango 2 totalmente covariante e Tv211 lo stesso tensore ma total-

mente controvariante (dove, se Tv2 è il tensore metrico covariante, Tv211 è il tensore metrico controvariante ossia la sua matrice inversa... DEVE essere la sua matrice inversa e questo fornisce un modo per controllare di avere fatto le trasformazioni di coordinate in modo corretto...)

```
Tn2 = Dot[ Transpose[ Jnv ], Dot[ Tv2, Jnv ] ];
Tn211= Dot[ Inverse[ Jnv ], Dot[ Tv211,Transpose[ Inverse[ Jnv ] ] ] ];
```

Usando la permutazione degli indici, dunque si deve fare per ogni indice lo scambio tra l'indice da trasformare ed il primo indice, poi la trasformazione dell'indice fatto diventare primo e poi rimettere al suo posto l'indice scambiandolo con quello che era il primo e che ritorna ad essere il primo. Ovviamente se l'indice è covariante lo si moltiplica per la trasposta della matrice jacobiana mentre se è controvariante lo si moltiplica per la matrice inversa della jacobiana. Questo è l'algoritmo da applicare a tensori di qualsiasi rango.