Help for: The TensorPack Package

TensorPack is a software package (that is simple to use) for the manipulation of tensors in index format in the Maple evironment. Tensorpack uses a string representation as input and provides functions for output in index form. It then extends the functionality to basic algebra of tensors, substitution, covariant differentiation, contraction, raising/lowering indices, symmetry functions and other accessory functions. The output can be merged with text in the Maple environment to create a full working document with embedded dynamic functionality.

This package is fully compatible with the Riemann & Canon Tensor Packages, and with the Maple Physics libraries (manipulation with tensor expressions).

Descriptions of user functions:

The exported functions in TensorPack are:

[*Absorbd*, *Absorbg*, *CDF*, *CDS*, *PrintArray*, *PrintSubArray*, *T*, *TEDS*, *TELS*, *TPset*, *anglesymm*, *antisymm*, *cod*, *contract*, *dotT*, *fpcodiff*, *raise*, *symm*
]

The functions can be divided into 10 main groups:
- **Output functions** - to output tensor expressions in covariant format: T
- **Algebraic substitution functions**: TEDS, TELS
- **Time differential functions**: dotT - outputs the time derivative of a tensor expression
- **Covariant differential functions:** cod - outputs the covariant derivative of a tensor expression
- **Symmetry functions**: symm, antisymm, anglesymm - outputs the expressions of tensor expressions after symmetry operations
- **Contraction function**: contract - forms the algebraic expression of the contraction of 2 indices
- **Metric & deltas absoption functions**: Absorbd, Absorbg
- **Flag functions**: CDS, CDF
- **Coordinate setting function**: TPset
- **Array functions**: PrintArray,PrintSubArray

Details of functions:

Output functions: T(expr) -output tensor expressions in covariant format. The main function of TensorPack is the **T** function

T (expr)
- expr = string indexed expression
- outputs a scalar, vector, tensor of derivative in covariant form

Examples:

```
> restart : with(Riemann) : with(Canon) : with(TensorPack) : CDF(0) : CDS(index) :
```

output a contravariant vector (using a positive index)

```
> expr[1] := u[a] : T(%);
```

$$u^{a} \qquad \textbf{(2.1.1)}$$

output a covariant vector (using a negative index)

```
> expr[2] := u[-a] : T(%);
```

$$u_{a} \tag{2.1.2}$$

output a contravariant, covariant and mixed tensors:

```
> expr[3] := X[a, b] : T(%);
```

$$X^{a\ b} \tag{2.1.3}$$

```
> expr[4] := Y[-c,-d] : T(%);
```

$$Y_{c\ d} \tag{2.1.4}$$

```
> expr[5] := R[-a, b, c, d] : T(%);
```

$$R_{a}^{\ b\ c\ d} \tag{2.1.5}$$

Tensor equations are formatted in the expected output for a linear function, eg.

```
> expr[5] := A[-a, b] = 2·B[-a, b] + C[-a, b] : T(%);
```

$$A_{a}^{\ b} = 2\,B_{a}^{\ b} + C_{a}^{\ b} \tag{2.1.6}$$

Algebraic substitution functions: TEDS(expr1, tensor), TELS(expr1,expr2)

TEDS (expr1,expr2)

- expr1 = in the form s=a, which is a tensor string equation with s=the term to be sustituted, and a=the term to replace s
- expr2 = an expression in which term s is DIRECTLY relaced by a (indices must be identical in s and a for the substitution to occur)

Note: only one substitution is carried out for each TEDS function

Examples:

```
> restart : with(Riemann) : with(Canon) : with(TensorPack) : CDF(0) : CDS(index) :
```

The following shows an example of a TEDS function:

```
> expr2 := A[-a, b] = 2·B[-a, b] + C[-a, b] : T(%); expr1 := C[-a, b] - 2·B[-a, b]
      = 0 : T(%); TEDS(expr1, expr2) : T(%);
```

$$A_{a}^{\ b} = 2\,B_{a}^{\ b} + C_{a}^{\ b}$$

$$C_{a}^{\ b} - 2\,B_{a}^{\ b} = 0$$

$$A_{a}^{\ b} = 2\,C_{a}^{\ b} \tag{2.2.1}$$

In the following example, there are 2 consective TEDS functions:

```
> expr2 := P[-a, a] = g[-a, a] + u[-a]·u[a] : T(%); expr1 := u[-a]·u[a] = -1 : T(%);
      expr3 := TEDS(expr1, expr2) : expr4 := g[-a, a] = 4 : T(%); TEDS(expr4,
      expr3) : T(%);
```

$$P_{a}^{\ a} = g_{a}^{\ a} + u_{a}\,u^{a}$$

$$u_{a}\,u^{a} = -1$$

$$g_{a}^{\ a} = 4$$

(2.2.2)

$$P_a{}^a = 3 \tag{2.2.2}$$

TELS (expr1,expr2)

• expr1 = in the form s=a, which is a tensor string equation

• expr2 = an expression in which term s is relaced by a (dummy indices may be swapped)

Note:

• only one substitution is carried out for each TELS function (it is hoped to be extended in updated versions of TensorPack)

• TELS is only partially functional, and applies only to single terms

Examples:

```
> restart : with(Riemann) : with(Canon) : with(TensorPack) : CDF(0) : CDS(index) :
```

The following shows an example of a TELS function:

```
> expr2 := X[-a, a] = u[-a]·u[a] : T(%); expr1 := u[-b]·u[b] = -1 : T(%); expr3
      := TELS(expr1, expr2) : T(%);
```

$$X_a{}^a = u_a u^a$$

$$u_b u^b = -1$$

$$X_a{}^a = -1 \tag{2.2.3}$$

TELS does not work for multiple terms:

```
> expr2 := P[-a, a] = g[-a, a] + u[-a]·u[a] + 2·u[-c]·u[c] : T(%); expr1 := u[-b]
     ·u[b] = -1 : T(%); expr3 := TELS(expr1, expr2) : T(%);
```

$$P_a{}^a = g_a{}^a + u_a u^a + 2\, u_c u^c$$

$$u_b u^b = -1$$

$$P_a{}^a = g_a{}^a + u_a u^a - 2 \tag{2.2.4}$$

• It is not recommended to use TELS in complex expressions in this package.

Time differential function: dotT(expr) - forms the time derivative of a tensor expression

dotT (expr)

• expr = string indexed expression

• outputs the time derivative of the tensor represented by expr

Note:

1. In TensorPack, we use the term `dot', immediately before the tensor, to indicate that the term is the time differetiative. i.e. dotA=d/dt(A)

2. dotT assumes the Liebniz rule for tensor products

Examples:

```
> restart : with(Riemann) : with(Canon) : with(TensorPack) : CDF(0) : CDS(index) :
```

The following shows examples of a dotT function, initially for a vector:

```
> expr := A[-a] : T(%); dotT(expr) : T(%);
```

$$A_{a}$$

$$dotA_{a} \tag{2.3.1}$$

for an expression (using the Liebniz rule):

```
> expr := A[-a]·B[b] : T(%); dotT(expr) : T(%);
```

$$A_{a} B^{b}$$

$$dotA_{a} B^{b} + A_{a} dotB^{b} \tag{2.3.2}$$

for an equation:

```
> expr := C[-a, b] = A[-a]·B[b] : T(%); dotT(expr) : T(%);
```

$$C_{a}{}^{b} = A_{a} B^{b}$$

$$dotC_{a}{}^{b} = dotA_{a} B^{b} + A_{a} dotB^{b} \tag{2.3.3}$$

```
> expr := X[a, b] = Y[a, b] + W[a]·Z[b] : T(%); dotT(expr) : T(%);
```

$$X^{a\ b} = Y^{a\ b} + W^{a} Z^{b}$$

$$dotX^{a\ b} = dotY^{a\ b} + dotW^{a} Z^{b} + W^{a} dotZ^{b} \tag{2.3.4}$$

Covariant differential functions: cod(expr,deriv) - forms the covariant derivative of a tensor expression

cod (expr,deriv)

- expr = string indexed expression
- deriv = the derivative
- outputs the derivative of the tensor expr with respect to deriveby expr

fpcodiff (expr)

- expr = string indexed expression, must be a derivative
- outputs the fully-project covariant derivative of the tensor expr with respect to deriveby expr

Note:

1. There are 2 output formats for derivatives:
   - Riemann (Portugal) (default)
   - indexed, using the semicolon as the indicator of differentiation

   See below for setting the format of the derivative output.
2. The derivative is input with an UPPERCASE index in the tensor term.

Examples:

The covariant differentiative of a tensor is expressed as:

```
> restart : with(Riemann) : with(Canon) : with(TensorPack) : CDF(0) : CDS(index) :
> CDS(index) : expr[13] := X[-a,-B] : T(%);
```

$$X_{a\,;b} \tag{2.4.1}$$

The output format of the expression requires a setup command `CDS(index)`:

```
> CDS(index);
```

Alternatively, the original Riemann package (Portugal) format can be the output if the setup comand is `CDS(Riemann)`:

```
> CDS(Riemann) : X[-a,-B] : T(%);
```

$$codiff_{-b}\left(X_{a}\right) \tag{2.4.2}$$

We prefer to use the index format as the output form. It is commonly ussed in the literature. Furthermore it can be used for multiple derivatives (there appears to be a bug in the Riemann program for this.)

```
> CDS(index) : expr[14] := X[-a,-B,-C] : T(%);
```

$$X_{a\,;b\,;c} \tag{2.4.3}$$

The cod function can be used for tensor equations, again using the Liebniz rule for products:

```
> expr := X[a, b] = Y[a, b] + W[a]·Z[b] : T(%); expr2 := cod(expr,-c) : T(%); expr3
      := cod(expr2,-d) : T(%);
```

$$X^{a\ b} = Y^{a\ b} + W^{a} Z^{b}$$

$$X^{a\ b}{}_{;c} = Y^{a\ b}{}_{;c} + W^{a}{}_{;c} Z^{b} + W^{a} Z^{b}{}_{;c}$$

$$X^{a\ b}{}_{;c\,;d} = Y^{a\ b}{}_{;c\,;d} + W^{a}{}_{;c\,;d} Z^{b} + W^{a}{}_{;c} Z^{b}{}_{;d} + W^{a}{}_{;d} Z^{b}{}_{;c} + W^{a} Z^{b}{}_{;c\,;d} \tag{2.4.4}$$

The fully projected covariant derivative is

```
> fpcodiff(u[a,-E]) : T(%);
```

$$P^{a}{}_{\_a} P_{\_e}{}^{e} u^{\_a}{}_{;e} \tag{2.4.5}$$

```
> fpcodiff(u[a, b,-c, d,-E]); T(%);
```

$$P_{a,\,-\_a} P_{b,\,-\_b} P_{-c,\,-\_c} P_{d,\,-\_d} P_{-\_e,\,e} u_{\_a,\,\_b,\,-\_c,\,\_d,\,-E}$$

$$P^{a}{}_{\_a} P^{b}{}_{\_b} P_{c}{}^{-\_c} P^{d}{}_{\_d} P_{\_e}{}^{e} u^{\_a\ \_b}{}_{\_c}{}^{\_d}{}_{;e} \tag{2.4.6}$$

```
>
```

```
> fpcodiff(u[a, b, c, d, E]) : T(%);
```

$$P^{a}{}_{\_a} P^{b}{}_{\_b} P^{c}{}_{\_c} P^{d}{}_{\_d} P^{\_e}{}_{e} u^{\_a\ \_b\ \_c\ \_d}{}_{;e} \tag{2.4.7}$$

```
> fpcodiff(u[a, b,-c, d,-E]) : T(%);
```

$$P^{a}{}_{\_a} P^{b}{}_{\_b} P_{c}{}^{-\_c} P^{d}{}_{\_d} P_{\_e}{}^{e} u^{\_a\ \_b}{}_{\_c}{}^{\_d}{}_{;e} \tag{2.4.8}$$

```
>
```

```
>
```

```
>
```

Symmetry functions: symm, antisymm, anglesymm - outputs the expressions of tensor expressions after symmetry operations

symm (expr,s,e)
antisymm (expr,s,e)

anglesymm (expr,s,t,e,v)

• expr = string indexed expression

• s = the index to start the index symmetrization

• e = the index to end the index symmetrization

• outputs the full set of terms resulting in the symmetry operation on the tensor expr:

symm: produces the symmetric terms of the expression

antisymm: produces the antisymmetric terms of the expression

anglesymm: produces orthogonal projections of vectors and the orthogonally projected symmetric trace-free part of tensors

Examples:

Symmetric and antisymmteric tensors can be fully expressed, for tensors with less than 5 indices:

```
> restart : with(Riemann) : with(Canon) : with(TensorPack) : CDF(0) : CDS(index) :
```

```
> symm(X[a, b, c], a, b) : T(%);
```

$$\frac{1}{2} X^{a\ b\ c} + \frac{1}{2} X^{b\ a\ c} \qquad \textbf{(2.5.1)}$$

```
> symm(X[a, b, c], a, c) : T(%);
```

$$\frac{1}{6} X^{a\ b\ c} + \frac{1}{6} X^{a\ c\ b} + \frac{1}{6} X^{b\ c\ a} + \frac{1}{6} X^{b\ a\ c} + \frac{1}{6} X^{c\ a\ b} + \frac{1}{6} X^{c\ b\ a} \qquad \textbf{(2.5.2)}$$

```
> antisymm(X[a, b, c], a, b) : T(%);
```

$$\frac{1}{2} X^{a\ b\ c} - \frac{1}{2} X^{b\ a\ c} \qquad \textbf{(2.5.3)}$$

```
> antisymm(X[a, b, c], a, c) : T(%);
```

$$\frac{1}{6} X^{a\ b\ c} - \frac{1}{6} X^{a\ c\ b} + \frac{1}{6} X^{b\ c\ a} - \frac{1}{6} X^{b\ a\ c} + \frac{1}{6} X^{c\ a\ b} - \frac{1}{6} X^{c\ b\ a} \qquad \textbf{(2.5.4)}$$

The symmetry functions apply only to the contravariant or covariant indices, depending on a and b, which must be of the same type.

```
> antisymm(X[a, -b, c], a, c) : T(%);
```

$$\frac{1}{2} X^{a}{}_{b}{}^{c} - \frac{1}{2} X^{c}{}_{b}{}^{a} \qquad \textbf{(2.5.5)}$$

For the anglesymm function:

For a vector:

```
> anglesymm(v[a], a, a, b, b) : T(%);
```

$$P^{a}{}_{b} v^{b} \qquad \textbf{(2.5.6)}$$

The use angle brackets denotes orthogonal projections of vectors and the orthogonally projected symmetric trace-free part of tensors (Ellis & van Elst, 1998)
Angle bracket symetry requires a special routine, *anglesymm*

For a tensor:

```
> anglesymm(v[a, b], a, b, c, d) : T(%);
```

(2.5.7)

$$\left(\frac{1}{2} P^{a}{}_{c} P^{b}{}_{d} + \frac{1}{2} P^{b}{}_{c} P^{a}{}_{d} - \frac{1}{3} P^{a\ b} P_{c\ d}\right) v^{c\ d} \qquad \textbf{(2.5.7)}$$

Contraction function: contract **-** forms the algebraic expression of the contraction of 2 indices

contract (a=b,expr)

- expr = string indexed expression
- a = the index to be contacted
- b = the index to replace a
- outputs the new expression containing the contracted indices

Examples:

```
> restart : with(Riemann) : with(Canon) : with(TensorPack) : CDF(0) : CDS(index) :
> expr := X[-a, b] + 2·Y[-a, b] + Z[-a, b] : T(%);
```

$$X_{a}{}^{b} + 2\,Y_{a}{}^{b} + Z_{a}{}^{b} \qquad \textbf{(2.6.1)}$$

```
> expr2 := contract(expr, a, b) : T(%);
```

$$X_{a}{}^{a} + 2\,Y_{a}{}^{a} + Z_{a}{}^{a} \qquad \textbf{(2.6.2)}$$

Contraction of tensor expressions involves simply the Maple substitution function:

```
> expr3 := expand(subs(b = a, expr)) : T(%);
```

$$X_{a}{}^{a} + 2\,Y_{a}{}^{a} + Z_{a}{}^{a} \qquad \textbf{(2.6.3)}$$

Note: The TEDS and TPset functions can be combined with contract to show contracted terms.

Metric & deltas absoption functions: Absorbg, Absorbd **-** absorbs `g` and kronecker delta (`δ`) terms into an expression

Absorbg (expr)
Absorbd(expr)

- expr = string indexed expression
- Absorbg outputs the new expression where the g term is absorbed
- Absorbd outputs the new expression where the kronecker delta (`δ`) term is absorbed

Tensor terms containing metric or kronecker delta terms can be simplified:

```
> restart : with(Riemann) : with(Canon) : with(TensorPack) : CDF(0) : CDS(index) :
> expr := delta[-a, b]·X[-b] + g[-b,-a]·X[b] : T(%);
```

$$\delta_{a}{}^{b} X_{b} + g_{b\,a} X^{b} \qquad \textbf{(2.7.1)}$$

```
> expr2 := Absorbd(expr) : T(%);
```

$$X_{a} + g_{b\,a} X^{b} \qquad \textbf{(2.7.2)}$$

```
> expr3 := Absorbg(expr2) : T(%);
```

$$2\,X_{a} \qquad \textbf{(2.7.3)}$$

These functions can often quickly simplify long expressions, eg.

```
> expr4 := antisymm(g[a, b]·X[-b, c], a, c) : T(%);
```

$$\frac{1}{6} g^{a\ b} X_b{}^c - \frac{1}{6} g^{a\ c} X_b{}^b + \frac{1}{6} g^{b\ c} X_b{}^a - \frac{1}{6} g^{b\ a} X_b{}^c + \frac{1}{6} g^{c\ a} X_b{}^b - \frac{1}{6} g^{c\ b} X_b{}^a \tag{2.7.4}$$

```
> expr5 := Absorbg(expr4) : T(%);
```

$$-\frac{1}{6} g^{a\ c} X_b{}^b + \frac{1}{6} g^{c\ a} X_b{}^b \tag{2.7.5}$$

```
> expr6 := TEDS(g[a, c] = g[c, a], expr5) : T(%);
```

$$0 \tag{2.7.6}$$

Flag functions: CDS, CDF- flag setting for output messages and derivatives

CDF (n)

• n=0 : no message output

• n<>0 (default): message are output, either as indicators, warnings or information

CDS (s)

• s=index : derivatives are expressed in index format using `;`

• s<>index : derivatives are expressed in Riemann (Portugal) format

Examples:

In the following example, message outputs depend on the value set in CDF:

```
> restart : with(Riemann) : with(Canon) : with(TensorPack) : CDF(0) : CDS(index) :
> expr := X[a] + Y[a] : T(%);
```

$$X^a + Y^a \tag{2.8.1}$$

```
> CDF(1) : TEDS(X[a] = Y[a], expr) : T(%);
```

DiagnosticsFlag changed to 1

" NEW EQUATION:"

$$2\ Y^a \tag{2.8.2}$$

```
> CDF(0) : TEDS(X[a] = Y[a], expr) : T(%);
```

$$2\ Y^a \tag{2.8.3}$$

In the following example, derivative outputs format depends on the value set in CDS:

```
> CDF(1) : CDS(riemann) :
```

DiagnosticsFlag changed to 1

codiff format changed to riemann **(2.8.4)**

```
> expr := X[a, -B] : T(%);
```

$$codiff_{-b}(X^a) \tag{2.8.5}$$

```
> CDS(index) : expr := X[a, -B] : T(%);
```

codiff format changed to index

$$X^{a}{}_{;b} \tag{2.8.6}$$

Coordinate setting function:TPset - prepares tensor expression for working in coordinates

TPset(expr)

• expr = string indexed expression

• returns the tensor in covariant format, and also prepares memory for allocation of components

Tensor pack is a algebraic string-based system (i.e. it represents tensor expressions with strings, but does not define a tensor entity in memory.) However, using TPset, the components of a tensor expressions term can be shown and worked with, if required, using the Riemann package. The (TensorPack) TPset function prepares the tensor expression for working with the component values of the tensor. For a complete list of coordiante functions for tensors, refer to Portugal & Sautú (1997) or the helpfile with the Riemann package. Some brief examples are shown below.

Examples:

In the following example, we work in a 2-dimensional (x-y) plane, which requires us to set the dimension and coordinates.

```
> restart : with(Riemann) : with(Canon) : with(TensorPack) : CDF(0) : CDS(index) :
> Dimension := 2; coordinates(x, y);
```

$$Dimension := 2$$

$$\text{The coordinates are:}$$

$$X^{1} = x$$

$$X^{2} = y \tag{2.9.1}$$

We define vectors A^{a} and B^{a}, and prepare them for components:

```
> expr := A[a] : TPset(expr); show(expr); expr := B[a] : TPset(expr); show(expr);
```

$$A^{a}$$

$$A^{x} = A^{x}$$

$$A^{y} = A^{y}$$

$$B^{a}$$

$$B^{x} = B^{x}$$

$$B^{y} = B^{y} \tag{2.9.2}$$

and set the component values:

```
> acomp(A[1] = 10·x); acomp(A[2] = -2·y); show(A[a]); acomp(B[1] = 15·y);
      acomp(B[2] = 3·x·x); show(B[a]);
```

$$A^{x} = 10\,x$$

$$A^{y} = -2\,y$$

$$B^{x} = 15\,y$$

(2.9.3)

$$B^{y} = 3\,x^{2} \qquad \textbf{(2.9.3)}$$

We define a new tensor based on the previous tensors; the components are also shown.

```
> expr := C[a] = A[a] + B[a] : TPset(expr); show(C[a]);
```

$$C^{a} = A^{a} + B^{a}$$

$$C^{x} = 10\,x + 15\,y$$

$$C^{y} = -2\,y + 3\,x^{2} \qquad \textbf{(2.9.4)}$$

This can be done for tensors of higher rank, and covariant dervatives.

```
>
```

Array functions: PrintArray,PrintSubArray **-** show list of expressions

PrintArray(arr) - prints a complete array as a set of tensor equations in covariant format
PrintSubArray(arr,s,l,f)- prints a complete array as a set of tensor equations in covariant format
arr = the array to be printed
s= the first equation to be printed
l= the last equation to be printed
f= flag to indicate if the index of each equation is also to be printed
• f=0 : no indices shown
• f<>0 : indices shown

Examples:

```
> restart : with(Riemann) : with(Canon) : with(TensorPack) : CDF(0) : CDS(index) :
> arr[1] := X[a] = Y[a] :
> arr[2] := Z[-b, C] = K[-b, C] :
> arr[3] := A[c] = B[-c] :
> PrintArray(arr);
```

$$X^{a} = Y^{a}$$

$$Z_{b}^{\;;c} = K_{b}^{\;;c}$$

$$A^{c} = B_{c} \qquad \textbf{(2.10.1)}$$

```
> PrintSubArray(arr, 2, 3, 1);
```

$$2,\ Z_{b}^{\;;c} = K_{b}^{\;;c}$$

$$3,\ A^{c} = B_{c} \qquad \textbf{(2.10.2)}$$

```
> PrintSubArray(arr, 1, 2, 0);
```

$$X^{a} = Y^{a}$$

$$Z_{b}^{\;;c} = K_{b}^{\;;c} \qquad \textbf{(2.10.3)}$$

```
>
```

Loading TensorPack

TensorPack requires that the Riemann, Canon and TensorPack be accessible to the worksheet. (See Installation and testing Instructions - below):

OPTION 1:

```
> restart : with(Riemann) : with(Canon) : with(TensorPack) : CDF(0) : CDS(index) :
```

OPTION 2:

```
> restart : with(Riemann) : with(Canon) : with(TensorPack) : CDF(0) : CDS(index) :
>libname := "C:/Riemann", libname;
>libname := "C:/Invar", libname;\011
>libname := "C:/TensorPack", libname;
>with(Riemann): with(Canon): with(TensorPack):
```

Installation and testing Instructions:

The TensorPack package is distributed free of charge. The authors kindly ask that researchers who have used the package cite references describing it.

The TensorPack package can be downloaded from:
http://www.deakin.edu.au/~peterhuf/
or
http://www.bach2roq.com/science/maths/GR/TensorPack.html

The Riemann and Canon packages (and installation instructions) are available at:
http://www.cbpf.br/~portugal/Riemann.html
http://www.lncc.br/~portugal/Invar.html

Installation instructions (TensorPack)
1. This software requires Maple 16 or above to be available. If this is not loaded, install it first.
2. Obtain the Riemann & Canon packages, and installation instructions, from
- http://www.cbpf.br/~portugal/Riemann.html
- http://www.lncc.br/~portugal/Invar.html

3. Load these Riemann & Canonpackages as instructed.
4. Loading TensorPack:

* OPTION 1: All files in one folder - this allows for the folder to be fully contained & self-sufficient
o TensorPack.mla, Riemann.mla, Canon.mla in one folder
o Open a Maple worksheet (.mw is the preferred format, although the software will run in .mws format)
o In the opening line of the worksheet write:
>with(Riemann): with(Canon): with(TensorPack):
The packages are now ready to use. See “Running & testing the program” below.

* OPTION 2: Create a package directory reference - this requires the system to reference the library
o Save the code and the help files in a directory. Let's say (the user may choose other paths)
o C:\Riemann (for the Riemann package)
o C:\Invar (for the Canon package)
o C:\TensorPack (for the TensorPack package)

- b. Start a Maple session. Issue the commands:

```
>libname := "C:/Riemann", libname;
>libname := "C:/Invar", libname;
>libname := "C:/TensorPack", libname;
>with(Riemann): with(Canon): with(TensorPack):
```

Running & testing the program
For help see the TensorPack-helpfile.mw
(Assuming option 1 or 2 is followed for installation above) Open the worksheet.
(If errors occur, give the command `restart` before trying again.)

Example:

> with(Riemann): with(Canon): with(TensorPack);
[Absorbd, Absorbg, CDF, CDS, PrintArray, PrintSubArray, T, TEDS, TELS, TPset, anglesymm, antisymm, cod, contract, dotT, fpcodiff, raise, symm]

#the array immediately above is the exported functions of TensorPack

> T(u[a,-b]);

$$u_{a}{}^{b}$$

Testing the program

TESTING OPTION 1 (QUICK OPTION):

1. Open a new worksheet in the folder containing the package
2. Type the following:
3.
> restart: with(Riemann): with(Canon): with(TensorPack): CDF(0): CDS(index):
> read “test-tensorpack.mpl”;
3. The file runs 20 tests with output “ok”, if all is working.

TESTING OPTION 2 (more complete check):

1. Open the worksheet “TensorPack-testoutput2.mw”.
2. Run this file (TensorPack-testoutput2.mw) by clicking the full compile button (!!!)
3. Compare to the correct output file (TensorPack-testoutput2.pdf)
4. For help see the helpfile (TensorPack-helpfile.mw)

Error reporting

For help or error reporting, please contact
Dr. Peter Huf (peterhuf@deakin.edu.au or peter@bach2roq.com)