The HyperKähler Potential for an Exceptional Next-to-Minimal Orbit

Piotr Kobak
Instytut Matematyki
Uniwersytet Jagielloński
ul. Reymonta 4
30-059 Krakow
Poland
E-mail: `kobak@im.uj.edu.pl`

Andrew Swann
Department of Mathematics and Computer Science
University of Southern Denmark
Odense University
Campusvej 55
DK-5230 Odense M
Denmark
E-mail: `swann@imada.sdu.dk`

Revision: 1.4 Last Modified: January 5, 2000

Contents

1 Introduction

The purpose of this file is to present some calculations done with the computer algebra system `Maple` needed for the paper (Kobak and Swann, 1998). However, in the process we developed some routines that may be of more general interest for working with the exceptional Lie algebra $\mathfrak{g}_2$.

The classification of simple Lie algebras over $\mathbb{C}$ gives four infinite families, known as the classical Lie algebras, and five exceptional examples. The smallest of these is the 14-dimensional algebra $\mathfrak{g}_2$. One definition of G_2 is as the automorphism group of the octonians $\mathbb{O}$ or Cayley numbers. The group acts preserving the 7-dimensional space of imaginary octonians and so has a natural description as a group of 7×7 matrices. It is this concrete description that we use here.

In many situations $\mathfrak{g}_2$ is very like a classical Lie algebra and not too hard to work with. However, in (Kobak and Swann, 1998), $\mathfrak{g}_2$ arose as one of only two exceptional cases in the problem we were considering. Surprisingly the other exceptional case was that of the classical algebra $\mathfrak{sl}(3, \mathbb{C})$, which we happened to have tackled first some years earlier in (Kobak and Swann, 1993).

The problem we were considering is the following. A complex Lie algebra contains some elements that are *nilpotent*, i.e., elements which become zero when raised to a high enough power. For example any matrix which is strictly upper triangular has this property. We may classify such matrices up to similarity, i.e., by their Jordan form. In an arbitrary Lie algebra this corresponds to determining the orbits of the adjoint action on such elements. Kronheimer (1990) showed that every such orbit admits a special Riemannian structure, known as a hyperKähler metric. Swann (1991) showed that Kronheimer's metrics had the special property of being given by a potential function on the orbit, a so-called *hyperKähler potential*. The problem now is to compute this function explicitly. In (Kobak and Swann, 1999) this was done for the simplest orbits and the next most simple were considered in (Kobak and Swann, 1998). Surprisingly in this latter paper there was a uniform expression for the potential across all simple Lie algebras apart from $\mathfrak{sl}(3, \mathbb{C})$ and $\mathfrak{g}_2^{\mathbb{C}}$. In the uniform case it was possible for us to do the computation by hand, but for $\mathfrak{g}_2^{\mathbb{C}}$ we resorted to using `Maple`.

This file provides specific routines to find the hyperKähler potential for the next-to-minimal nilpotent orbit in $\mathfrak{g}_2^{\mathbb{C}}$ and also general routines for working with $\mathfrak{g}_2$ as 7×7 matrices. The general routines are to be found in §2. The specific calculations, more supporting code and `Maple` output are in §3.

Most of this work was carried at the University of Bath and we are grateful to the EPSRC of Great Britain and the KBN in Poland for financial support.

1.1 About this Document

This document is a literate program in the sense of Knuth (1992). One source file combines both `Maple` code and documentation which can be typeset using LaTeX. The documented version divides the code up in to manageable chunks each numbered and with an accompanying description of its function. This file is written in `noweb` which is available from `http://www.cs.virginia.edu/~nr/noweb/`.

To extract the `Maple` source, either enter

```
noweb -t g2.nw
```

or for more control

```
notangle -Rg2-gen g2.nw > g2-gen
notangle -Rg2nmin-direct g2.nw > g2nmin-direct
notangle -Rg2direct-sample g2.nw > g2direct-sample
```

This will produce maple files `g2-gen`, `g2nmin-direct`, `g2direct-sample`. `g2-gen` contains general routines for working with a presentation of $\mathfrak{g}_2$ as 7×7 matrices. The other two files are specific to the problem of determining the hyperKähler potential of the next-to-minimal nilpotent orbit in $\mathfrak{g}_2^{\mathbb{C}}$.

To get the documentation, type

```
noweave -delay -index g2.nw > g2.tex
```

and then process the resulting file with either LaTeX or pdfLaTeX.

2 An Embedding of the Exceptional Group

This section describes the file `g2-gen` which contains general routines for dealing with the exceptional Lie group G_2. The code is built around an embedding of $\mathfrak{g}_2$ in the orthogonal algebra $\mathfrak{so}(7)$.

The structure of `g2-gen` is as follows:

1 ⟨*g2-gen* 1⟩≡

```
⟨Top matter g2-gen 2⟩
⟨SO7 matrices 4⟩
⟨Lie bracket 6⟩
⟨Inner product 9⟩
⟨Real structure 10⟩
⟨G2 embedding 11⟩
```

Each part is described in a separate section below.

2.1 Top Matter

We start with a header comment identifying this file.

2 ⟨*Top matter* `g2-gen` 2⟩≡ (1) 3▷

```
# g2-gen
# Maple code for calculating in the exceptional Lie algebra G2
# via the embedding in SO(7)
# by
# Piotr Kobak and Andrew Swann
#
# This code is generated from a noweb source file g2.nw
# See that for further description and comments.
# RCS info from g2.nw:
# $Id: g2.nw,v 1.4 2000/01/05 14:10:18 swann Exp $
```

For the calculations, we use matrix operators from the package `linalg`.

3 ⟨*Top matter* `g2-gen` 2⟩+≡ (1) ◁2

```
with(linalg):
```

2.2 Orthogonal Matrices

We take $\mathfrak{so}(7,\mathbb{C})$ to be the set of (7×7) matrices X preserving a non-degenerate symmetric matrix B, i.e.,

$$X^t B + BX = 0.$$

The standard choice for B is just the identity matrix, but for us a better choice is the antidiagonal matrix

$$B = \begin{pmatrix} & 0 & & & & & 1 \\ & & & & & 1 & \\ & & & & 1 & & \\ & & & 1 & & & \\ & & 1 & & & & \\ & 1 & & & & 0 & \\ 1 & & & & & & \end{pmatrix}.$$

Let us call this matrix AntiDiagonal.

4 ⟨*SO7 matrices* 4⟩≡ (1) 5▷

```
AntiDiagonal := matrix(7,7,0):
for i to 7 do
    AntiDiagonal[i,8-i] := 1;
od:
```

Defines:
`AntiDiagonal`, never used.

Elements of $\mathfrak{so}(7,\mathbb{C})$ now satisfy

$$x_{i,j} = -x_{8-j,8-i}. \tag{2.1}$$

In `Maple` matrices of a special form may be specified by using an index function. We set up such a function `` `index/so7` `` for elements of $\mathfrak{so}(7,\mathbb{C})$ in 2.7. This may be used to create elements of $\mathfrak{so}(7,\mathbb{C})$ by beginning with the output of `so7matrix()` and modifying entries. We also provide sparse matrices by `so7sparse()` which have unassigned entries equal to zero.

5 ⟨*SO7 matrices* 4⟩+≡ (1) ◁4

```
⟨SO7 index function 14⟩
so7matrix := proc() array(so7,1..7,1..7) end:
so7sparse := proc() array(so7,sparse,1..7,1..7) end:
```

Defines:
`so7matrix`, never used.
`so7sparse`, used in chunks 11–13.

Uses `so7` 14.

2.3 Lie Brackets

What makes $\mathfrak{so}(7,\mathbb{C})$ into a Lie algebra is the presence of a Lie bracket $[X,Y]$. This is given in terms of matrix multiplication by

$$[X,Y]=XY-YX.$$

However, for some of our purposes this can be slow to compute. If we are prepared to assume that our matrices are in $\mathfrak{so}(7,\mathbb{C})$, then a saving is to be had by knowing that the result is also an element of $\mathfrak{so}(7,\mathbb{C})$.

For elements of $\mathfrak{so}(7,\mathbb{C})$, we have

$$X=-BX^tB,$$

since $B^2=1$. Thus $YX=B(XY)^tB$ and we may now write the Lie bracket for such matrices as

$$[X,Y]=XY-B(XY)^tB=XY-(XY)^\dagger,$$

where we define $A^\dagger=-BA^tX$. This has the advantage that it only involves one matrix multiplication of X and Y. If $A=(a_{ij})$, then

$$(A^\dagger)_{ij}=a_{8-j,8-i}.$$

Let us provide this operation as `so7transpose`, thinking of this as analogue of the transpose operation adapted to our bilinear form.

6 ⟨*Lie bracket 6*⟩≡ (1) 7▷

```
so7transpose := proc(A)
    local a,i,j,out;
    a:=evalm(A);
    out:=matrix(7,7);
    for i from 1 to 7 do
        for j from 1 to 7 do
            out[i,j] := A[8-j,8-i];
        od;
    od;
    evalm(out);
end:
```

Defines:
`so7transpose`, used in chunk 8.

We provide the function `LieBracket` which when given two arguments uses the naïve definition, but which also accepts an optional third argument giving a way to compute YX from XY.

7 ⟨*Lie bracket* 6⟩+≡ (1) ◁6 8▷

```
LieBracket := proc (X, Y, trans)
    local Z;
    if (nargs = 3) then
        Z := evalm(X &* Y);
        RETURN(evalm(Z - trans(Z)));
    elif (nargs = 2) then
        RETURN(evalm(X &* Y - Y &* X));
    else
        ERROR(`Wrong number of arguments to`, procname);
    fi;
end:
```

Defines:
`LieBracket`, used in chunk 8.
Uses `X` 21.

We then define the Lie bracket in $\mathfrak{so}(7, \mathbb{C})$ by

8 ⟨*Lie bracket* 6⟩+≡ (1) ◁7

```
so7Lb := proc(X,Y)
    LieBracket(X,Y,so7transpose);
end:
```

Defines:
`so7Lb`, used in chunks 23, 24, 26, 36, 39, and 59.
Uses `LieBracket` 7, `so7transpose` 6, and `X` 21.

2.4 Inner Products

The natural inner product on a matrix Lie algebra is a multiple of $-\operatorname{Tr}(XY)$, minus the trace of the product of matrices. We call that multiple `MetricNormalisation` and treat this as a global variable. Doing a matrix multiplication here is inefficient, instead we use

$$\operatorname{Tr}(XY) = \sum_{i,j} x_{ij} y_{ji}.$$

We let the user optionally tell the function the sizes of the matrices involved. When this argument is provided we assume the user knows what they are doing and do not make any error checks.

9 ⟨*Inner product 9*⟩≡ (1)

```
MetricForm := proc(X,Y,nn::integer)
    local boundsx,boundsy,i,j,n,total;
    global MetricNormalisation;
    if (nargs = 3) then
        n := nn;
    else
        boundsx := [op(2,evalm(X))];
        boundsy := [op(2,evalm(Y))];
        n := op([1,2],boundsx);
        if not(n=op([2,2],boundsx))
            or not(n=op([1,2],boundsy))
            or not(n=op([2,2],boundsy))
        then
            ERROR(`Arguments of`, procname,
            `need to be square matrices of the same size`);
        fi;
    fi;
    total := 0;
    for i to n do
        for j to n do
            total := total + X[i,j]*Y[j,i];
        od;
    od;
    - MetricNormalisation * total;
end:
```

Defines:
`MetricForm`, used in chunks 26 and 29.
Uses `X` 21.

2.5 Real Structures

Associated to the complex Lie algebra $\mathfrak{so}(7,\mathbb{C})$ is a real Lie algebra $\mathfrak{so}(7)$. In fact there are many different embeddings of $\mathfrak{so}(7)$ in $\mathfrak{so}(7,\mathbb{C})$; the way to pick one out is to specify a real structure σ. A real structure is like a conjugation operation: it is a conjugate linear function and squares to the identity. In the presence of our bilinear form B, the standard definition for σ is

$$\sigma(X) = B\overline{X}B.$$

However, for elements of $\mathfrak{so}(7,\mathbb{C})$, the right-hand side is simply $-\overline{X}^t$. In our computations we will sometimes only want to apply this operation to matrices X whose entries are real. We therefore divide this definition into two parts.

10 ⟨*Real structure* 10⟩≡ (1)

```
RConj := proc(X)
    evalm(- transpose(X));
end:
Conj := proc(X)
    map(conjugate,evalm(RConj(X)));
end:
```

Defines:
`Conj`, used in chunk 26.
`RConj`, used in chunks 24, 26, and 29.
Uses `X` 21.

2.6 The Embedding

The algebra $\mathfrak{g}_2^{\mathbb{C}}$ is the subalgebra of $\mathfrak{so}(7,\mathbb{C})$ which preserves a particular three-form φ. This three-form encodes Cayley multiplication on the octonians $\mathbb{O}$. In fact a generic three-form will have the property that its stabiliser is isomorphic to $\mathfrak{g}_2$. However, we need to make a choice that is compatible with the bilinear form B. This compatibility is expressed by

$$(v^tBw)\,\mathrm{vol} = 6(v \lrcorner \varphi) \wedge (w \lrcorner \varphi) \wedge \varphi,$$

where vol is the standard volume form. Labelling our standard basis of the dual of $\mathbb{C}^7$ by $e_1, \dots, e_7$, we may take

$$\varphi = e_1e_4e_7 + e_2e_4e_6 + e_3e_4e_5 - \sqrt{2}(e_1e_2e_3 + e_5e_6e_7),$$

where $abc := a \wedge b \wedge c$. Writing $g(v,w) = v^tBw$, the complexification of Cayley multiplication is given by

$$g(v.w,u) = \varphi(v,w,u).$$

A convenient way to understand $\mathfrak{g}_2$ is via one of its subalgebras. Consider the action of G_2 on $\mathbb{O}$. For any unit vector $v \in \mathbb{O}$, we have a splitting $\mathbb{O} = \mathbb{R} \oplus W$, where $\mathbb{R}$ is spanned by v and W is the orthogonal complement. Now v acts on W via Cayley multiplication and has the property that v^2 acts as -1. Thus W may be thought of as a complex vector space, necessarily of dimension 3. The stabiliser of v in G_2 is isomorphic to $SU(3)$. Passing to Lie algebras and complexifying, this means that $\mathfrak{g}_2^{\mathbb{C}}$ has a subalgebra isomorphic to $\mathfrak{sl}(3,\mathbb{C})$, the algebra of (3×3)-matrices A such that $\operatorname{Tr} A = 0$. Under the action of $\mathfrak{sl}(3,\mathbb{C})$, we have the decomposition

$$\mathfrak{g}_2^{\mathbb{C}} = \mathfrak{sl}(3,\mathbb{C}) \oplus V^{1,0} \oplus V^{0,1}, \tag{2.2}$$

where $V^{1,0}$ is the usual three-dimensional representation of $\mathfrak{sl}(3,\mathbb{C})$ on $\mathbb{C}^3$ given by matrices multiplying column vectors and $V^{0,1}$ is the dual of $V^{0,1}$. Our embedding of $\mathfrak{g}_2^{\mathbb{C}}$ is based on (2.2). For elements of $\mathfrak{sl}(3,\mathbb{C})$ itself, this is easy: we map

$$A \mapsto \begin{pmatrix} A & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -A^t \end{pmatrix}.$$

The following routine assumes that the matrix A is trace-free.

11 ⟨*G2 embedding* 11⟩≡ (1) 12▷

```
sl3 := proc(A)
    local Am,i,j,out;
    Am := convert(eval(A),matrix);
    if (not([op(2,eval(Am))]=[1..3,1..3])) then
        ERROR(‘Argument of‘, procname,
            ‘must be convertible to a 3x3 matrix‘);
    fi;
    if not(trace(Am)=0) then
        print(‘Warning: matrix passed to sl3 is not trace-free‘);
    fi;
    out := so7sparse();
    for i from 1 to 3 do
        for j from 1 to 3 do
            out[i,j]:= Am[i,j];
        od;
    od;
    evalm(out);
```

```
  end:
```

Defines:
`sl3`, used in chunks 21, 36, 39, and 59.
Uses `so7sparse` 5.

The embeddings of $V^{1,0}$ and $V^{0,1}$ are determined by the three-form φ, once one realises that $W \otimes \mathbb{C} \cong V^{1,0} \oplus V^{0,1}$. Alternatively, one may use the splitting $\mathbb{C}^7 = \mathbb{C} \oplus W \otimes \mathbb{C}$ and the fact that $\mathfrak{so}(7, \mathbb{C}) \cong \Lambda^2\mathbb{C}^7$. This implies

$$\begin{aligned} \mathfrak{so}(7, \mathbb{C}) &\cong \Lambda^2(\mathbb{C} \oplus V^{1,0} \oplus V^{0,1}) \\ &= V^{1,0} \oplus V^{0,1} \oplus \Lambda^2 V^{1,0} \oplus \Lambda^2 V^{0,1} \oplus V^{1,0} \otimes V^{0,1} \\ &= 2V^{1,0} \oplus 2V^{0,1} \oplus \mathfrak{sl}(3, \mathbb{C}) \oplus \mathbb{C}, \end{aligned}$$

since $\Lambda^2 V^{1,0} \cong V^{0,1}$. In matrix terms this implies that the two copies of $V^{1,0}$ are

$$\begin{pmatrix} 0 & 0 & 0 & x & 0 & 0 & 0 \\ 0 & 0 & 0 & y & 0 & 0 & 0 \\ 0 & 0 & 0 & z & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -z & -y & -x \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ a & b & 0 & 0 & 0 & 0 & 0 \\ c & 0 & -b & 0 & 0 & 0 & 0 \\ 0 & -c & -a & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Examining the action of $\mathfrak{sl}(3, \mathbb{C})$ on these matrices, we see that $a \leftrightarrow y$, $b \leftrightarrow x$ and $c \leftrightarrow z$. It is natural to take a, b, c to be essentially the same multiples of y, x, z, but the condition that $[V^{1,0}, V^{1,0}] = V^{0,1}$ gives $a = -ky$, $b = kx$ and $c = kz$, for some constant k. If we require $V^{0,1}$ to consist of elements related also by the same factor k, then one finds $k^2 = 2$. Thus For $V^{1,0}$ we have

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} 0 & 0 & 0 & x & 0 & 0 & 0 \\ 0 & 0 & 0 & y & 0 & 0 & 0 \\ 0 & 0 & 0 & z & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -z & -y & -x \\ y/\sqrt{2} & -x/\sqrt{2} & 0 & 0 & 0 & 0 & 0 \\ -z/\sqrt{2} & 0 & x/\sqrt{2} & 0 & 0 & 0 & 0 \\ 0 & z/\sqrt{2} & -y/\sqrt{2} & 0 & 0 & 0 & 0 \end{pmatrix}.$$

12 ⟨*G2 embedding* 11⟩+≡ (1) ◁11 13▷

```
  V10 := proc(L::list)
      local i,out;
      if not(nops(L)=3) then
          ERROR(`Argument to`, procname, `must have 3 elements`);
      fi;
      out := so7sparse();
      out[6,3] :=  L[1]/sqrt(2);
```

```
    out[5,1] :=  L[2]/sqrt(2);
    out[7,2] :=  L[3]/sqrt(2);
    for i from 1 to 3 do
        out[i,4] := L[i];
    od;
    evalm(out);
end:
```

Defines:
`V10`, used in chunks 21 and 59.
Uses `so7sparse` 5.

Dually for $V^{0,1}$, the embedding is

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} 0 & 0 & 0 & 0 & y/\sqrt{2} & -x/\sqrt{2} & 0 \\ 0 & 0 & 0 & 0 & -z/\sqrt{2} & 0 & x/\sqrt{2} \\ 0 & 0 & 0 & 0 & 0 & z/\sqrt{2} & -y/\sqrt{2} \\ x & y & z & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -z & 0 & 0 & 0 \\ 0 & 0 & 0 & -y & 0 & 0 & 0 \\ 0 & 0 & 0 & -x & 0 & 0 & 0 \end{pmatrix}.$$

13 ⟨*G2 embedding* 11⟩+≡ (1) ◁12

```
V01 := proc(L::list)
    local i,out;
    if not(nops(L)=3) then
        ERROR(`Argument to`, procname, `must have 3 elements`);
    fi;
    out := so7sparse();
    out[3,6] :=  L[1]/sqrt(2);
    out[1,5] :=  L[2]/sqrt(2);
    out[2,7] :=  L[3]/sqrt(2);
    for i from 1 to 3 do
        out[4,i] := L[i];
       od;
    evalm(out);
end:
```

Defines:
`V01`, used in chunk 59.
Uses `so7sparse` 5.

2.7 An Index Function for Orthogonal Matrices

The index function `` `index/so7` `` will be used by `Maple` in two ways depending on how many arguments it is given. For two arguments, the entry is simply being evaluated, and we just need to return the value. This will be 0 for the values on the antidiagonal. For three arguments, the entry is being assigned.

14 ⟨*SO7 index function* 14⟩≡ (5)

```
`index/so7` := proc(inds,A,v)
     ⟨set local variables for SO7 index function 15⟩
     if (nargs = 2) then
        ⟨read SO7 entry 16⟩
     elif (nargs = 3) then
        ⟨set SO7 entry 17⟩
     else ERROR(`Invalid arguments passed to`, procname);
     fi;
end:
```

Defines:
`so7`, used in chunk 5.

We ensure that values are only given to entries above the antidiagonal. The antidiagonal is conveniently specified by those entries x_{ij} of X such that $i + j = 8$ and those above the antidiagonal are given by $i + j < 8$. If x_{ij} lies below the antidiagonal we get/put information from the corresponding position above the antidiagonal according to (2.1). We set `t` to be $8 - i - j$, so the sign of `t` tells us whether we are below, on or above the antidiagonal.

15 ⟨*set local variables for SO7 index function* 15⟩≡ (14)

```
local ind,i,j,t,swap;
ind := [op(inds)];
i := ind[1];
j := ind[2];
t := i+j-8;
if t>0 then
  swap := i;
  i:=8-j;
  j:=8-swap;
fi;
```

16 ⟨*read SO7 entry* 16⟩≡ (14)

```
if t=0 then
   RETURN(0);
elif t<0 then
   RETURN(A[i,j]);
else
   RETURN(-A[i,j]);
fi;
```

17 ⟨*set SO7 entry* 17⟩≡ (14)

```
if t=0 then
    if op(v)=0 then
        RETURN(0);
    else
        ERROR(`Elements of SO(7) cannot have`,
        `non-zero entries on the antidiagonal`)
    fi;
elif t<0 then
    A[i,j] := op(v);
else
    A[i,j] := -op(v);
fi;
```

3 Direct Computation of a HyperKähler Potential

The exceptional Lie algebra $\mathfrak{g}_2^{\mathbb{C}}$ has four orbits of nilpotent elements. The smallest of these is the associated bundle of the Wolf space $G_2 / SO(4)$ and so is generally fairly well understood. The next biggest orbit $\mathcal{O}_{\text{nmin}}$ is of cohomogeneity two for the action of the compact group G_2, so one can hope to calculate the invariant hyperKähler potentials for this orbit directly. In (Kobak and Swann, 1998) we showed that this came done to showing that a certain endomorphism J is an almost complex structure, i.e., that it squares to -1, on the tangent space of the orbit.

We divide this material up in to a support file `g2nmin-direct` and the actual computation `g2direct-sample`.

3.1 Support File `g2nmin-direct`

18 ⟨*g2nmin-direct* 18⟩≡
⟨*Top matter* `g2nmin-direct` 19⟩
⟨*Base point* 21⟩
⟨*Almost complex structure* 22⟩
⟨*Test J-squared* 27⟩

3.1.1 Top Matter

First we identify this file.

19 ⟨*Top matter* `g2nmin-direct` 19⟩≡ (18) 20▷

```
# g2nmin-direct
# Maple code for a direct computation of hyperKaehler potentials
# for the next-to-minimal nilpotent orbit in G2
# by
# Piotr Kobak and Andrew Swann
#
# This code is generated from a noweb source file g2.nw
# See that for further description and comments.
# RCS info from g2.nw:
# $Id: g2.nw,v 1.4 2000/01/05 14:10:18 swann Exp $
```

We need the file `g2-gen`.

20 ⟨*Top matter* `g2nmin-direct` 19⟩+≡ (18) ◁19

```
read ‘g2-gen‘;
```

3.1.2 Base Point

Now define our base point X in $\mathcal{O}_{\text{nmin}}$. As the orbit is of cohomogeneity two, there are two real parameters `s` and `t`, which we can take to be strictly positive. If either of them is zero then X becomes a point of the minimal orbit instead.

21 ⟨*Base point* 21⟩≡ (18)

```
print(‘Our base point is‘);
X := evalm( sl3([[0,s,0],[0,0,0],[0,0,0]])
              + V10([t*sqrt(2),0,0]) );
```

Defines:
`X`, used in chunks 7–10, 23–26, 29, 36, 39, and 59.
Uses `sl3` 11 and `V10` 12.

3.1.3 The Almost Complex Structure

A typical tangent vector to the orbit at X has the form $\xi_A = [A, X]$, where $A \in \mathfrak{g}_2^{\mathbb{C}}$. The formula for the candidate almost complex structure J as given in our paper is

$$\begin{aligned} J\xi_A = {} & -2\rho_1[X, \sigma\xi_A] \\ & + 4\rho_2(2[X,[\sigma X,[X,\sigma\xi_A]]] - [X,[X,[\sigma X,\sigma\xi_A]]]) \\ & - 2\rho_{11}\left\langle \sigma\xi_A, X\right\rangle [X,\sigma X] \\ & + 4\rho_{12}\Big(\left\langle \sigma\xi_A, [X,[\sigma X, X]]\right\rangle [X,\sigma X] \\ & \qquad\qquad + \left\langle \sigma\xi_A, X\right\rangle [X,[\sigma X,[X,\sigma X]]]\Big) \\ & - 8\rho_{22}\left\langle \sigma\xi_A, [X,[\sigma X,X]]\right\rangle [X,[\sigma X,[X,\sigma X]]]. \end{aligned} \tag{3.1}$$

Here ρ is the hyperKähler potential, and ρ_i etc. are partial derivatives with respect to two natural local coordinates η_1 and η_2 defined by the Lie algebra, see §3.1.5 below.

We need to optimise this formula for J for the implementation, otherwise `Maple` just seems to die in the computations. The important thing to do is to assign quantities only involving X to global variables. This means that they do not have to repeatedly calculated in the routine for J. The quantities involving ξ_A however depend on A and should be local variables.

22 ⟨*Almost complex structure* 22⟩≡ (18)

```
print(`Defining J`);
⟨Global parts of J 24⟩
⟨Procedure for J including local parts of J 25⟩
```

Uses `J` 25.

It is also possible to optimise the formula (3.1) in another way. The coefficient of ρ_2 was rewritten during our proof, and the original form

$$4\Big([X,[\sigma\xi_A,[X,\sigma X]]] + [X,[\sigma X,[X,\sigma\xi_A]]]\Big)$$

is better here as it involves more global variables.

23 ⟨*Formula for J* 23⟩≡ (25)

```
evalm(-2*rho1*XsXiA
      +4*rho2*so7Lb(X, evalm(so7Lb(sXiA,XsX)+so7Lb(sX,XsXiA)))
      -2*(rho11*IpsXiAX - 2*rho12*IpsXiAXsXX) * XsX
      +4*(rho12*IpsXiAX - 2*rho22*IpsXiAXsXX) * XsXXsX );
```

Uses `so7Lb` 8, `sX` 24, `X` 21, `XsX` 24, and `XsXXsX` 24.

The global quantities are

$$\begin{aligned} \texttt{sX} &:= \sigma X \\ \texttt{XsX} &:= [X,\sigma X] \\ \texttt{XsXX} &:= [X,[\sigma X,X]] \\ \texttt{XsXXsX} &:= [X,[\sigma X,[X,\sigma X]]] \end{aligned}$$

Note that we may assume the entries of `X` are real and so use `RConj` here.

24 ⟨*Global parts of J* 24⟩≡ (22)

```
sX  := RConj(X):
XsX  := so7Lb(X,sX):
XsXX  := evalm(-so7Lb(X,XsX)):
XsXXsX  := so7Lb(X,so7Lb(sX,XsX)):
```

Defines:
- `sX`, used in chunks 23 and 25.
- `XsX`, used in chunks 23, 25, and 29.
- `XsXX`, used in chunks 25 and 26.
- `XsXXsX`, used in chunks 23 and 25.

Uses `RConj` 10, `so7Lb` 8, and `X` 21.

The quantities we can split off locally are

$$\begin{aligned}\texttt{sXiA} &:= \sigma\xi_A\\ \texttt{XsXiA} &:= [X,\sigma\xi_A]\\ \texttt{IpsXiAX} &:= \langle\sigma\xi_A, X\rangle\\ \texttt{IpsXiAXsXX} &:= \langle\sigma\xi_A,[X,[\sigma X,X]]\rangle\ .\end{aligned}$$

If `J` is given a second argument, of any value, then the first argument is assumed to have real entries and we can use `RConj` instead of `Conj`.

25 ⟨*Procedure for J including local parts of J* 25⟩≡ (22)

```
J := proc(XiA,r)
    local sXiA, XsXiA, IpsXiAX, IpsXiAXsXX;
    global X, sX, XsX, XsXX, XsXXsX, rho1, rho2, rho11, rho12, rho22;
    ⟨Local parts of J 26⟩
    ⟨Formula for J 23⟩
end:
```

Defines:
  `J`, used in chunks 22 and 27.
Uses `sX` 24, `X` 21, `XsX` 24, `XsXX` 24, and `XsXXsX` 24.

26 ⟨*Local parts of J* 26⟩≡ (25)

```
    if (nargs = 1) then
        sXiA := Conj(XiA);
    else
        sXiA := RConj(XiA);
    fi;
    XsXiA := so7Lb(X,sXiA);
    IpsXiAX := MetricForm(sXiA,X,7);
    IpsXiAXsXX := MetricForm(sXiA,XsXX,7);
```

Uses `Conj` 10, `MetricForm` 9, `RConj` 10, `so7Lb` 8, `X` 21, and `XsXX` 24.

3.1.4 Testing for an Almost Complex Structure

We provide various routines for finding when $J^2 = -1$. Each has an optional second argument which if present indicates that the first is real.

Firstly, we test via a simple computation of $J^2\xi_A + \xi_A$, which one would like to vanish.

27 ⟨*Test J-squared* 27⟩≡ (18) 28 ▷

```
J2P := proc(xA,r)
    if (nargs = 1) then
        evalm(J(J(xA))+xA);
    else
        evalm(J(J(xA,1),1)+xA);
    fi;
end:
```

Defines:
  `J2P`, used in chunk 28.
Uses `J` 25.

Secondly, one can change variables in the derivatives to those with respect to s and t instead of η_1 and η_2.

It is tempting to automatically simplify the expression $J^2\xi_A + \xi_A$, but as these are 7×7 matrices that would be very time consuming. We let the user do that when they want to. However, mapping `expand` on to the entries can help alot.

28 ⟨*Test J-squared* 27⟩+≡ (18) ◁ 27

```
⟨Change variables 29⟩
J2stP := proc(xA,r)
    local out;
    global dsdt;
    if (nargs = 1) then
        out := subs(dsdt,J2P(xA));
    else
        out := subs(dsdt,J2P(xA,1));
    fi;
    map(expand,out);
end:
```

Defines:
  `J2stP`, used in chunks 37, 40, and 59.
Uses `dsdt` 30 and `J2P` 27.

3.1.5 Change of Variables

The two main invariants are $\eta_1 = \langle X, \sigma X \rangle$ and $\eta_2 = \langle Y, \sigma Y \rangle$, with $Y = [X, \sigma X]$. We need to know how these are related to the variables s and t in the definition of X and be able to convert corresponding partial derivatives.

29 ⟨*Change variables* 29⟩≡ (28) 30▷

```
print(`Computing change of variables`);
eta1 := simplify(MetricForm(X,RConj(X),7));
eta2 := simplify(-MetricForm(XsX,XsX,7));
```

Defines:
`eta1`, used in chunks 31, 32, and 57.
`eta2`, used in chunks 31, 32, and 57.
Uses `MetricForm` 9, `RConj` 10, `X` 21, and `XsX` 24.

Let `rhos` and `rhot` be the partial derivatives of $\rho(s,t)$ with respect to s and t. Similarly, write `rho1` and `rho2` for the derivatives of $\rho(\eta_1,\eta_2)$ with respect to η_1 and η_2. `D1rhost` will hold the expressions for ρ_1 and ρ_2 in terms of ρ_s and ρ_t. Similarly `D2rhost` will contain the second derivatives. These two lots of rules are collected in `dsdt`.

30 ⟨*Change variables* 29⟩+≡ (28) ◁29

```
⟨Compute first derivatives D1rhost 31⟩
⟨Compute second derivatives D1rhost 32⟩
dsdt := D1rhost union D2rhost:
```

Defines:
`dsdt`, used in chunk 28.
Uses `D1rhost` 31.

We use the chain rule. First we compute abstract partial derivatives of a function of two variables `z1` and `z2` each of which is itself a function of s and t. We try to ensure that the answers are written purely in terms of the partial derivatives ρ_s, ρ_t, ρ_1 and ρ_2.

31 ⟨*Compute first derivatives* `D1rhost` 31⟩≡ (30)

```
D1rho := {  rhos=diff( rho(z1(s,t),z2(s,t)),s),
            rhot=diff( rho(z1(s,t),z2(s,t)),t)}:
D1rho := subs( { D[1](rho)(z1(s,t),z2(s,t))=rho1,
          D[2](rho)(z1(s,t),z2(s,t))=rho2
        }, D1rho ):
D1rho := simplify(subs({z1(s,t)=eta1,z2(s,t)=eta2}, D1rho)):
D1rhost:=simplify(solve(D1rho,{rho1,rho2})):
```

Defines:
`D1rhost`, used in chunks 30 and 32.
Uses `eta1` 29 and `eta2` 29.

The computation of the second derivatives is similar.

32 ⟨*Compute second derivatives* `D1rhost` 32⟩≡ (30)

```
D2rho := { rhoss=diff(rho(z1(s,t),z2(s,t)),s,s),
           rhost=diff(rho(z1(s,t),z2(s,t)),s,t),
           rhott=diff(rho(z1(s,t),z2(s,t)),t,t)
          }:
D2rho := subs({ D[1](rho)(z1(s,t),z2(s,t))=rho1,
                D[2](rho)(z1(s,t),z2(s,t))=rho2,
                D[1,2](rho)(z1(s,t),z2(s,t))=rho12,
                D[2,1](rho)(z1(s,t),z2(s,t))=rho12,
                D[1,1](rho)(z1(s,t),z2(s,t))=rho11,
                D[2,2](rho)(z1(s,t),z2(s,t))=rho22
              }, D2rho):
D2rho := simplify(subs({z1(s,t)=eta1,z2(s,t)=eta2}, D2rho)):
D2rho := subs(D1rhost,D2rho):
D2rhost:=simplify(solve(D2rho,{rho11,rho12,rho22})):
```

Uses `D1rhost` 31, `eta1` 29, and `eta2` 29.

3.2 Sample Computation

In this section we present the actual computation of the potential for the next-to-minimal nilpotent orbit of $\mathfrak{g}_2^{\mathbb{C}}$. We include output from the `Maple` session in §3.2.5.

33 ⟨*g2direct-sample* 33⟩≡

⟨*Top matter for* `g2direct-sample` 34⟩
⟨*Initialisation for* `g2direct-sample` 35⟩
⟨*First Main Equation* 36⟩
⟨*Second Main Equation* 39⟩
⟨*Solutions* 47⟩

34 ⟨*Top matter for* `g2direct-sample` 34⟩≡ (33)

```
# g2direct-sample
# Maple code example of a direct computation of the hyperKaehler
# potential for the next-to-minimal nilpotent orbit in G2
# by
# Piotr Kobak and Andrew Swann
#
```

```
# This code is generated from a noweb source file g2.nw
# See that for further description and comments.
# RCS info from g2.nw:
# $Id: g2.nw,v 1.4 2000/01/05 14:10:18 swann Exp $
```

3.2.1 Initialisation

First we read in the support macros defined above, having made sure that `Maple` is its virgin state. We put `MetricNormalisation` to be `k^2` to save space in the output.

35 ⟨*Initialisation for* `g2direct-sample` 35⟩≡ (33)

```
restart;
MetricNormalisation:=k^2;
read `g2nmin-direct`;
```

3.2.2 First Main Equation

We obtain equations for ρ and its partial derivatives by enforcing the condition $J^2\xi_A = -\xi_A$ for good choices of ξ_A. The first equation comes by considering an element in $\mathfrak{sl}(3,\mathbb{C})$.

36 ⟨*First Main Equation* 36⟩≡ (33) 37▷

```
A := matrix(3,3,0):
A[2,3] := 1/s:
print(A);
xA := so7Lb(X,sl3(A));
```

Uses `sl3` 11, `so7Lb` 8, and `X` 21.

Now compute $J^2\xi_A + \xi_A$.

37 ⟨*First Main Equation* 36⟩+≡ (33) ◁36 38▷

```
A1 := J2stP(xA,1);
```

Defines:
`A1`, used in chunk 38.
Uses `J2stP` 28.

We see that there is only one non-zero entry above the antidiagonal. This gives us our first equation.

38 ⟨*First Main Equation 36*⟩+≡ (33) ◁37

```
print(‘First equation‘);
e1 := -numer(A1[1,3]);
```

Defines:
`e1`, used in chunks 49, 50, and 53.
Uses `A1` 37.

```
                2                            4
 e1 := rhos   s + t rhot rhos - 4 k   s
```

3.2.3 Second Main Equations

Our second equation is obtained in the same way as the first, but we start with a different element of $\mathfrak{sl}(3,\mathbb{C})$. We reuse the variables `A` and `xA`.

39 ⟨*Second Main Equation 39*⟩≡ (33) 40▷

```
A := matrix(3,3,0):
A[1,1]:=1:
A[3,3]:=-1:
print(A);
xA := so7Lb(X,sl3(A));
```

Uses `sl3` 11, `so7Lb` 8, and `X` 21.

This time $J^2\xi_A + \xi_A$ is rather more complicated.

40 ⟨*Second Main Equation 39*⟩+≡ (33) ◁39 41▷

```
A2 := J2stP(xA,1);
```

Defines:
`A2`, used in chunk 41.
Uses `J2stP` 28.

We extract three equations from the numerators of the $(1,2)$, $(1,4)$ and $(2,7)$ entries. The second of these has a common factor of $t\sqrt{2}$ which we take out.

41 ⟨*Second Main Equation 39*⟩+≡ (33) ◁40 42▷

```
A2e := [numer(A2[1,2]),
        numer(A2[1,4]/t/sqrt(2)),
        numer(A2[2,7])];
```

Defines:
`A2e`, used in chunk 42.
Uses `A2` 40.

These three equations are linear in the second derivatives ρ_{ss}, ρ_{st} and ρ_{tt}. We will try to eliminate as many of these terms as we can. First we rearrange `A2e` so that we can get to the coefficients of these elements.

42 ⟨*Second Main Equation* 39⟩+≡ (33) ◁41 43▷

```
A2c := map(collect,A2e,{rhoss,rhost,rhott},distribute,factor);
```

Defines:
`A2c`, used in chunk 43.
Uses `A2e` 41.

Now eliminate ρ_{ss}.

43 ⟨*Second Main Equation* 39⟩+≡ (33) ◁42 44▷

```
A2e2 := [ coeff(A2c[2],rhoss)*A2c[1] - coeff(A2c[1],rhoss)*A2c[2],
          coeff(A2c[3],rhoss)*A2c[1] - coeff(A2c[1],rhoss)*A2c[3] ];
```

Uses `A2c` 42.

Collect second derivatives again and then eliminate ρ_{st}.

44 ⟨*Second Main Equation* 39⟩+≡ (33) ◁43 45▷

```
A2c2 := map(collect,A2e2,{rhoss,rhost,rhott},distribute,factor);
```

Defines:
`A2c2`, used in chunk 45.

45 ⟨*Second Main Equation* 39⟩+≡ (33) ◁44 46▷

```
A2e3 := expand(coeff(A2c2[2],rhost)*A2c2[1]
                  - coeff(A2c2[1],rhost)*A2c2[2]);
```

Defines:
`A2e3`, used in chunk 46.
Uses `A2c2` 44.

The pleasant surprise now is that there is no ρ_{tt} term left. Our second main equation is `A2e3` tidied up.

46 ⟨*Second Main Equation* 39⟩+≡ (33) ◁45

```
print(`Second equation`);
e2 := factor(A2e3);
```

Defines:
`e2`, used in chunks 47, 48, and 53.
Uses `A2e3` 45.

```
          4  2  3                                        2
e2 := 24 k  t  s  (2 rhos s + t rhot) (9 t rhos - rhot s)
```

3.2.4 Solutions

Equation e2 is so simple that we can solve it directly. We get either $\rho_t = -2s\rho_s/t$ or $\rho_t = 9t\rho_s/s$.

47 ⟨*Solutions* 47⟩≡ (33) 48▷

```
print(`Solutions to e2`);
sollist := [ -2*s*rhos/t, 9*t*rhos/s ];
```

Defines:
`sollist`, used in chunks 48–50.
Uses `e2` 46.

Let us first verify that these are indeed solutions to e2.

48 ⟨*Solutions* 47⟩+≡ (33) ◁47 49▷

```
print(`These two expressions should give zero`);
subs(rhot=sollist[1],e2);
subs(rhot=sollist[2],e2);
```

Uses `e2` 46 and `sollist` 47.

Which of these now give solutions to e1?

49 ⟨*Solutions* 47⟩+≡ (33) ◁48 50▷

```
print(`Substitute first solution in e1`);
factor(subs(rhot=op(1,sollist),e1));
```

Uses `e1` 38 and `sollist` 47.

This gives

```
         2      4
-s (rhos  + 4 k )
```

which has no real solutions.

However, we have more joy with the second element of `sollist`.

50 ⟨*Solutions* 47⟩+≡ (33) ◁49 51▷

```
print(`Substitute second solution in e1`);
s2 := simplify(subs(rhot=op(2,sollist),e1));
```

Defines:
`s2`, used in chunk 51.
Uses `e1` 38 and `sollist` 47.

51 ⟨*Solutions* 47⟩+≡ (33) ◁50 52▷

```
s3 := collect(numer(s2),rhos);
```

Defines:
`s3`, used in chunk 53.
Uses `s2` 50.

```
        2      2      2      4  2
s3 := (s  + 9 t ) rhos  - 4 k  s
```

Thus $\rho_s = 2k^2s/\sqrt{s^2+9t^2}$ and $\rho_t = 18k^2t/\sqrt{s^2+9t^2}$. Let us verify this.

52 ⟨*Solutions* 47⟩+≡ (33) ◁51 53▷

```
print(`Solutions for rhos and rhot are`);
solrhos := 2*k^2*s/sqrt(s^2+9*t^2);
solrhot := 18*k^2*t/sqrt(s^2+9*t^2);
```

Defines:
`solrhos`, used in chunks 53, 54, and 56.
`solrhot`, used in chunks 53 and 55.

53 ⟨*Solutions* 47⟩+≡ (33) ◁52 54▷

```
print(`The following three expressions should be zero`);
subs(rhos=solrhos,s3);
simplify(subs([rhos=solrhos,rhot=solrhot],e1));
subs([rhos=solrhos,rhot=solrhot],e2);
```

Uses `e1` 38, `e2` 46, `s3` 51, `solrhos` 52, and `solrhot` 52.

Integrating the equations for ρ_s and ρ_t gives

54 ⟨*Solutions* 47⟩+≡ (33) ◁53 55▷

```
int(solrhos,s)+fun1(t);
```

Uses `solrhos` 52.

```
    2      2 1/2  2
2 (s  + 9 t )    k  + fun1(t)
```

55 ⟨*Solutions* 47⟩+≡ (33) ◁54 56▷

```
int(solrhot,t)+fun2(s);
```

Uses `solrhot` 52.

```
    2      2 1/2  2
2 (s  + 9 t )    k  + fun2(s)
```

Equating these two expressions shows that `fun1` and `fun2` are constant. But ρ is only defined up to an additive constant, so

56 ⟨*Solutions* 47⟩+≡ (33) ◁55 57▷

```
solrho := int(solrhos,s);
```

Defines:
`solrho`, used in chunks 57 and 58.
Uses `solrhos` 52.

```
              2      2 1/2  2
solrho := 2 (s  + 9 t )    k
```

The solution in terms of η_1 and η_2 is claimed to be

$$k\sqrt{2}\sqrt{\eta_1 + \sqrt{6}\sqrt{\eta_1^2 - k^2\eta_2}}.$$

We can now check that this is indeed our solution.

57 ⟨*Solutions* 47⟩+≡ (33) ◁56 58▷

```
print(`The following should be zero`);
radsimp(k*sqrt(2*(eta1+sqrt(6*(eta1^2-k^2*eta2)))))
-solrho;
```

Uses `eta1` 29, `eta2` 29, and `solrho` 56.

It now remains to verify that `solrho` satisfies all the equations from $J^2 = -1$, even those we have not used. It seems simplest to work through a basis for $\mathfrak{g}_2$.

First set the derivatives to be those given by the solution.

58 ⟨*Solutions* 47⟩+≡ (33) ◁57 59▷

```
rhos:=diff(solrho,s);
rhot:=diff(solrho,t);
rhoss:=diff(rhos,s);
rhost:=diff(rhos,t);
rhott:=diff(rhot,t);
```

Uses `solrho` 56.

Now work through a basis of $\mathfrak{g}_2$, starting with $\mathfrak{sl}(3,\mathbb{C})$ and then each of the three dimensional spaces. Don't be particularly subtle!

59 ⟨*Solutions* 47⟩+≡ (33) ◁58

```
print(`All the remaining matrices should be zero`);
for i from 1 to 3 do
  for j from 1 to 3 do
  A := matrix(3,3,0);
  A[i,j]:=1;
  if i=j then A[3,3]:=-A[i,i] fi;
  xA := so7Lb(X,sl3(A));
  print(map(simplify,J2stP(xA,1)));
  od;
od;
```

```
  for i from 1 to 3 do
    v:=[0,0,0];
    v[i]:=1;
    xA := so7Lb(X,V10(v));
    print(map(simplify,J2stP(xA,1)));
    xA := so7Lb(X,V01(v));
    print(map(simplify,J2stP(xA,1)));
  od;
```

Uses `J2stP` 28, `sl3` 11, `so7Lb` 8, `V01` 13, `V10` 12, and `X` 21.

3.2.5 Maple Output

```
    |\^/|     Maple V Release 4 (University of Bath)
._|\|   |/|_. Copyright (c) 1981-1996 by Waterloo Maple Inc. All rights
 \  MAPLE  /  reserved. Maple and Maple V are registered trademarks of
 <____ ____>  Waterloo Maple Inc.
      |       Type ? for help.
# g2direct-sample
# Maple code example of a direct computation of the hyperKaehler
# potential for the next-to-minimal nilpotent orbit in G2
# by
# Piotr Kobak and Andrew Swann
#
# This code is generated from a noweb source file g2.nw
# See that for further description and comments.
# RCS info from g2.nw:
# $Id: mapleoutput-g2direct-sample,v 1.1 2000/01/05 14:11:32 swann Exp $
> restart;
> MetricNormalisation:=k^2;
                                                   2
                          MetricNormalisation := k

> read `g2nmin-direct`;
Warning, new definition for norm
Warning, new definition for trace
                              Our base point is

                    [                  1/2                ]
                    [0    s     0    t 2      0    0      0   ]
                    [                                         ]
                    [0    0     0      0      0    0      0   ]
                    [                                         ]
                    [0    0     0      0      0    0      0   ]
                    [                                         ]
               X := [                                     1/2]
                    [0    0     0      0      0    0    -t 2   ]
                    [                                         ]
                    [0    -t    0      0      0    0      0   ]
                    [                                         ]
                    [0    0     t      0      0    0     -s   ]
                    [                                         ]
                    [0    0     0      0      0    0      0   ]

                                  Defining J

bytes used=1000068, alloc=786288, time=0.78
```

```
                         Computing change of variables

                                       2   2      2
                            eta1 := 2 k  (s  + 3 t )

                                  2   4      2  2      4
                       eta2 := 4 k  (s  + 6 s  t  + 3 t )

bytes used=2000484, alloc=1179432, time=1.76
bytes used=3000676, alloc=1441528, time=2.69
> A := matrix(3,3,0):
> A[2,3] := 1/s:
> print(A);
                                [0    0     0 ]
                                [             ]
                                [0    0    1/s]
                                [             ]
                                [0    0     0 ]

> xA := so7Lb(X,sl3(A));
                          [0    0    1    0    0    0     0]
                          [                                ]
                          [0    0    0    0    0    0     0]
                          [                                ]
                          [0    0    0    0    0    0     0]
                          [                                ]
                    xA := [0    0    0    0    0    0     0]
                          [                                ]
                          [0    0    0    0    0    0    -1]
                          [                                ]
                          [0    0    0    0    0    0     0]
                          [                                ]
                          [0    0    0    0    0    0     0]

> A1 := J2stP(xA,1);
bytes used=4000988, alloc=1965720, time=4.02
bytes used=5001208, alloc=1965720, time=5.47
             [                  2                                      ]
             [              rhos        t rhot rhos                    ]
             [0 , 0 , - 1/4 ----- - 1/4 ----------- + 1 , 0 , 0 , 0 , 0]
             [                4             4                          ]
             [               k             k  s                        ]
             [                                                         ]
             [0 ,       0 ,       0 ,      0 ,      0 ,      0 ,      0]
             [                                                         ]
```

```
             [0 ,       0 ,       0 ,      0 ,      0 ,      0 ,      0]
             [                                                         ]
       A1 := [0 ,       0 ,       0 ,      0 ,      0 ,      0 ,      0]
             [                                                         ]
             [                                  2                      ]
             [                              rhos        t rhot rhos    ]
             [0 ,  0 ,  0 , 0 , 0 , 0 , 1/4 ----- + 1/4 ----------- - 1]
             [                                4             4          ]
             [                               k             k  s        ]
             [                                                         ]
             [0 ,       0 ,       0 ,      0 ,      0 ,      0 ,      0]
             [                                                         ]
             [0 ,       0 ,       0 ,      0 ,      0 ,      0 ,      0]

> print(`First equation`);
                                 First equation

> e1 := -numer(A1[1,3]);
                                2                      4
                      e1 := rhos  s + t rhot rhos - 4 k  s

> A := matrix(3,3,0):
> A[1,1]:=1:
> A[3,3]:=-1:
> print(A);
                                 [1    0     0]
                                 [            ]
                                 [0    0     0]
                                 [            ]
                                 [0    0    -1]

> xA := so7Lb(X,sl3(A));
                    [                     1/2                    ]
                    [0    -s    0     -t 2      0    0      0   ]
                    [                                            ]
                    [0    0     0        0      0    0      0   ]
                    [                                            ]
                    [0    0     0        0      0    0      0   ]
                    [                                            ]
              xA := [                                         1/2]
                    [0    0     0        0      0    0    t 2   ]
                    [                                            ]
                    [0    t     0        0      0    0      0   ]
                    [                                            ]
                    [0    0     -t       0      0    0      s   ]
```

```
                    [                                            ]
                    [0    0     0        0       0    0      0   ]

> A2 := J2stP(xA,1);
bytes used=6001516, alloc=1965720, time=7.01
bytes used=7001792, alloc=1965720, time=8.44
bytes used=8002112, alloc=1965720, time=9.93
bytes used=9110140, alloc=2162292, time=14.47
A2 :=

    [              2                             2
    [        s rhos        t s rhot rhoss       t  rhot rhost       t rhot rhos
    [0 , 1/4 ------- + 1/8 -------------- + 1/8 ------------- + 1/4 -----------
    [           4                 4                   4                  4
    [          k                 k                   k                  k

            2                                            2
           s  rhos rhoss       t s rhos rhost           t  rhos rhott
     + 1/4 ------------- + 3/8 -------------- - s + 1/8 ------------- , 0 ,
                 4                    4                       4
                k                    k                       k

                         2  1/2             2  1/2                 1/2
        1/2        t rhot  2          t rhos  2          t s rhos 2    rhoss
    -t 2    + 1/36 ------------ + 1/4 ------------ + 1/4 -------------------
                         4                  4                     4
                        k                  k                     k

            2       1/2                        1/2
           t  rhos 2    rhost        t s rhot 2    rhost
     + 1/4 ------------------ + 1/36 -------------------
                    4                         4
                   k                         k

             2       1/2                         2         2
            t  rhot 2    rhott             s rhot         t  rhos rhoss
     + 1/36 ------------------ , 0 , 1/144 ------- - 3/16 -------------
                     4                        4                 4
                    k                        k                 k

                                   2
            t s rhot rhoss        t  rhot rhost        t rhot rhos
     - 1/48 -------------- - 1/48 ------------- - 1/24 -----------
                   4                    4                   4
                  k                    k                   k
```

```
                                2                    3
        t s rhos rhost         s  rhot rhost        t  rhos rhost
 - 1/48 -------------- + 1/144 ------------- - 3/16 -------------
               4                     4                    4
              k                     k                    k  s

         2     2                                2               ]
        t  rhos          t s rhot rhott        t  rhos rhott    ]
 - 3/16 -------- + 1/144 -------------- - 1/48 ------------- , 0]
           4                    4                    4          ]
          k  s                 k                    k           ]

[                                      2         2
[                               s rhot          t  rhos rhoss
[0 , 0 , 0 , 0 , 0 , 0 , - 1/144 ------- + 3/16 -------------
[                                   4                 4
[                                  k                 k

                               2
        t s rhot rhoss        t  rhot rhost        t rhot rhos
 + 1/48 -------------- + 1/48 ------------- + 1/24 -----------
               4                    4                   4
              k                    k                   k

                                2                    3
        t s rhos rhost         s  rhot rhost        t  rhos rhost
 + 1/48 -------------- - 1/144 ------------- + 3/16 -------------
               4                     4                    4
              k                     k                    k  s

         2     2                                2           ]
        t  rhos          t s rhot rhott        t  rhos rhott]
 + 3/16 -------- - 1/144 -------------- + 1/48 -------------]
           4                    4                    4      ]
          k  s                 k                    k       ]

[0 , 0 , 0 , 0 , 0 , 0 , 0]

[              2       1/2                        1/2
[             t  rhot 2    rhost        t s rhos 2    rhost
[0 , 0 , 1/48 ------------------ + 1/48 -------------------
[                      4                         4
[                     k                         k
```

```
                1/2                       1/2
        t rhot 2    rhos        t s rhot 2    rhoss
 + 1/24 ---------------- + 1/48 -------------------
                4                        4
               k                        k

          2       1/2               2       1/2
         s  rhot 2    rhost        t  rhos 2    rhoss
 - 1/144 ------------------ + 3/16 ------------------
                  4                         4
                 k                         k

         3       1/2               2     2  1/2                   1/2
        t  rhos 2    rhost        t  rhos  2            t s rhot 2    rhott
 + 3/16 ------------------ + 3/16 ------------- - 1/144 -------------------
                4                      4                         4
               k  s                   k  s                      k

         2       1/2                     2  1/2
        t  rhos 2    rhott         s rhot  2                     1/2
 + 1/48 ------------------ - 1/144 ------------ , 0 , 0 , 0 , t 2
                 4                       4
                k                       k

              2  1/2             2  1/2                 1/2
        t rhot  2          t rhos  2          t s rhos 2    rhoss
 - 1/36 ------------ - 1/4 ------------ - 1/4 -------------------
              4                  4                     4
             k                  k                     k

        2       1/2                        1/2
       t  rhos 2    rhost        t s rhot 2    rhost
 - 1/4 ------------------ - 1/36 -------------------
                4                         4
               k                         k

         2       1/2      ]
        t  rhot 2    rhott]
 - 1/36 ------------------]
                 4        ]
                k         ]

[                   2                                  2
[             t rhot        t s rhos rhoss       t rhos
[0 , t - 1/36 ------- - 1/4 -------------- - 1/4 -------
```

```
[                4                 4                4
[               k                 k                k

                              2                    2
        t s rhot rhost       t  rhos rhost        t  rhot rhott
 - 1/36 -------------- - 1/4 ------------- - 1/36 ------------- , 0 ,
               4                   4                    4
              k                   k                    k

        2       1/2                        1/2
       t  rhot 2    rhost        t s rhos 2    rhost
- 1/48 ------------------ - 1/48 -------------------
                4                         4
               k                         k

                1/2                       1/2
        t rhot 2    rhos        t s rhot 2    rhoss
 - 1/24 ---------------- - 1/48 -------------------
                4                        4
               k                        k

          2       1/2               2       1/2
         s  rhot 2    rhost        t  rhos 2    rhoss
 + 1/144 ------------------ - 3/16 ------------------
                  4                         4
                 k                         k

         3       1/2               2     2  1/2                   1/2
        t  rhos 2    rhost        t  rhos  2            t s rhot 2    rhott
 - 3/16 ------------------ - 3/16 ------------ + 1/144 -------------------
                4                      4                         4
               k  s                   k  s                      k

         2       1/2                     2  1/2            ]
        t  rhos 2    rhott         s rhot  2               ]
 - 1/48 ------------------ + 1/144 ------------ , 0 , 0 , 0]
                 4                       4                 ]
                k                       k                  ]

[                        2                                  2
[                  t rhot        t s rhos rhoss       t rhos
[0 , 0 , -t + 1/36 ------- + 1/4 -------------- + 1/4 -------
[                     4                 4                4
[                    k                 k                k
```

```
                                  2                    2
            t s rhot rhost       t  rhos rhost        t  rhot rhott
     + 1/36 -------------- + 1/4 ------------- + 1/36 ------------- , 0 , 0 , 0
                   4                   4                    4
                  k                   k                    k

                  2                             2
            s rhos        t s rhot rhoss       t  rhot rhost       t rhot rhos
    , - 1/4 ------- - 1/8 -------------- - 1/8 ------------- - 1/4 -----------
               4                 4                   4                  4
              k                 k                   k                  k

            2                                            2           ]
           s  rhos rhoss       t s rhos rhost           t  rhos rhott]
     - 1/4 ------------- - 3/8 -------------- + s - 1/8 -------------]
                 4                    4                       4      ]
                k                    k                       k       ]

    [0 , 0 , 0 , 0 , 0 , 0 , 0]

> A2e := [numer(A2[1,2]),
>         numer(A2[1,4]/t/sqrt(2)),
>         numer(A2[2,7])];
              2                       2
A2e := [2 rhos  s + t s rhot rhoss + t  rhot rhost + 2 t rhot rhos

          2                                    4      2                       4
     + 2 s  rhos rhoss + 3 t s rhos rhost - 8 k  s + t  rhos rhott, -2 t (36 k

           2         2
     - rhot  - 9 rhos  - 9 s rhos rhoss - 9 t rhos rhost - s rhot rhost

                        2     2       2                     2
     - t rhot rhott), -s  rhot  + 27 t  rhos rhoss s + 3 t s  rhot rhoss

          2                                       2               3
     + 3 t  rhot rhost s + 6 t rhot rhos s + 3 t s  rhos rhost - s  rhot rhost

           3                  2     2      2                 2
     + 27 t  rhos rhost + 27 t  rhos  - t s  rhot rhott + 3 t  rhos rhott s]

> A2c := map(collect,A2e,{rhoss,rhost,rhott},distribute,factor);
                                       2
A2c := [t (3 s rhos + t rhot) rhost + t  rhos rhott
```

```
                                           2        4
     + s (2 s rhos + t rhot) rhoss + 2 rhos  s - 8 k  s + 2 t rhot rhos,

                                       2
    2 t (9 t rhos + rhot s) rhost + 2 t  rhot rhott + 18 t s rhos rhoss

                4       2         2
     - 2 t (36 k  - rhot  - 9 rhos ),

        2                   3      2           3
    (3 t  rhot s + 27 rhos t  + 3 s  t rhos - s  rhot) rhost

     + t s (-rhot s + 3 t rhos) rhott + 3 t s (9 t rhos + rhot s) rhoss

     + (9 t rhos - rhot s) (3 t rhos + rhot s)]

> A2e2 := [ coeff(A2c[2],rhoss)*A2c[1] - coeff(A2c[1],rhoss)*A2c[2],
>           coeff(A2c[3],rhoss)*A2c[1] - coeff(A2c[1],rhoss)*A2c[3] ];
                                                     2
A2e2 := [18 t s rhos (t (3 s rhos + t rhot) rhost + t  rhos rhott + s %1 rhoss

             2        4
     + 2 rhos  s - 8 k  s + 2 t rhot rhos) - s %1 (

                                       2
    2 t (9 t rhos + rhot s) rhost + 2 t  rhot rhott + 18 t s rhos rhoss

                4       2         2
     - 2 t (36 k  - rhot  - 9 rhos )), 3 t s (9 t rhos + rhot s) (

                                   2                                 2
    t (3 s rhos + t rhot) rhost + t  rhos rhott + s %1 rhoss + 2 rhos  s

          4
     - 8 k  s + 2 t rhot rhos) - s %1 (

        2                   3      2           3
    (3 t  rhot s + 27 rhos t  + 3 s  t rhos - s  rhot) rhost

     + t s (-rhot s + 3 t rhos) rhott + 3 t s (9 t rhos + rhot s) rhoss

     + (9 t rhos - rhot s) (3 t rhos + rhot s))]

%1 := 2 s rhos + t rhot
```

```
> A2c2 := map(collect,A2e2,{rhoss,rhost,rhott},distribute,factor);
              2                           2         2
A2c2 := [2 t s  (-2 rhot rhos s + 9 t rhos  - t rhot ) rhost

          2                             2         2
     + 2 t  s (-2 rhot rhos s + 9 t rhos  - t rhot ) rhott

                                          2           2       4
     + 2 t rhot s (-2 rhot rhos s - t rhot  + 9 t rhos  + 36 k  t),

     2      3     2           2  2      3                2     2
    s  (27 t  rhos  - 6 t rhos  s  + 2 s  rhos rhot + t s  rhot ) rhost

                3     2           2  2      3                2     2
     + s t (27 t  rhos  - 6 t rhos  s  + 2 s  rhos rhot + t s  rhot ) rhott - s

            2       4       3          2      2     2              2         4
    (216 s t  rhos k  - 27 t  rhot rhos  + 6 s  rhos  t rhot + 24 s  t rhot k

          3          2         3  2
     - 2 s  rhos rhot  - t rhot  s )]

> A2e3 := expand(coeff(A2c2[2],rhost)*A2c2[1]
>                 - coeff(A2c2[1],rhost)*A2c2[2]);
              4  4     3  4        4  4     2       4        5  3     2       4
A2e3 := 3888 s  t  rhos  k  - 432 s  t  rhot  rhos k  - 864 s  t  rhos  rhot k

           6          2  2  4       5  3     3  4         3  5     2       4
     + 48 s  rhos rhot  t  k  + 24 s  t  rhot  k  + 1944 s  t  rhos  rhot k

> print(`Second equation`);
                                Second equation

> e2 := factor(A2e3);
                     4  2  3                                         2
           e2 := 24 k  t  s  (2 s rhos + t rhot) (9 t rhos - rhot s)

> print(`Solutions to e2`);
                                Solutions to e2

> sollist := [ -2*s*rhos/t, 9*t*rhos/s ];
                                       s rhos    t rhos
                        sollist := [-2 ------, 9 ------]
                                         t         s
```

```
> print(‘These two expressions should give zero‘);
                     These two expressions should give zero

bytes used=10112140, alloc=2227816, time=15.89
> subs(rhot=sollist[1],e2);
                                       0

> subs(rhot=sollist[2],e2);
                                       0

> print(‘Substitute first solution in e1‘);
                        Substitute first solution in e1

> factor(subs(rhot=op(1,sollist),e1));
                                       2      4
                               -s (rhos  + 4 k )

> print(‘Substitute second solution in e1‘);
                        Substitute second solution in e1

> s2 := simplify(subs(rhot=op(2,sollist),e1));
                                 2  2      2     2      4  2
                            -rhos  s  - 9 t  rhos  + 4 k  s
                    s2 := - --------------------------------
                                           s

> s3 := collect(numer(s2),rhos);
                               2      2      2      4  2
                       s3 := (s  + 9 t ) rhos  - 4 k  s

> print(‘Solutions for rhos and rhot are‘);
                        Solutions for rhos and rhot are

> solrhos := 2*k^2*s/sqrt(s^2+9*t^2);
                                             2
                                            k  s
                          solrhos := 2 --------------
                                         2      2 1/2
                                       (s  + 9 t )

> solrhot := 18*k^2*t/sqrt(s^2+9*t^2);
                                              2
                                             k  t
                          solrhot := 18 --------------
                                          2      2 1/2
```

```
                                        (s  + 9 t )

> print(`The following three expressions should be zero`);
                 The following three expressions should be zero

> subs(rhos=solrhos,s3);
                                       0

> simplify(subs([rhos=solrhos,rhot=solrhot],e1));
                                       0

> subs([rhos=solrhos,rhot=solrhot],e2);
                                       0

> int(solrhos,s)+fun1(t);
                             2      2 1/2  2
                         2 (s  + 9 t )    k  + fun1(t)

> int(solrhot,t)+fun2(s);
                             2      2 1/2  2
                         2 (s  + 9 t )    k  + fun2(s)

> solrho := int(solrhos,s);
                                       2      2 1/2  2
                         solrho := 2 (s  + 9 t )    k

> print(`The following should be zero`);
                          The following should be zero

> radsimp(k*sqrt(2*(eta1+sqrt(6*(eta1^2-k^2*eta2)))))
> -solrho;
                                       0

> rhos:=diff(solrho,s);
                                            2
                                           k  s
                            rhos := 2 --------------
                                          2      2 1/2
                                      (s  + 9 t )

> rhot:=diff(solrho,t);
                                            2
                                           k  t
                           rhot := 18 --------------
                                          2      2 1/2
```

```
                                      (s  + 9 t )

> rhoss:=diff(rhos,s);
                                    2                2  2
                                   k                k  s
                  rhoss := 2 -------------- - 2 --------------
                                 2      2 1/2       2      2 3/2
                               (s  + 9 t )        (s  + 9 t )

> rhost:=diff(rhos,t);
                                            2
                                           k  s t
                          rhost := -18 --------------
                                         2      2 3/2
                                       (s  + 9 t )

> rhott:=diff(rhot,t);
                                   2                  2  2
                                  k                  k  t
                rhott := 18 -------------- - 162 --------------
                              2      2 1/2         2      2 3/2
                            (s  + 9 t )          (s  + 9 t )

> print(`All the remaining matrices should be zero`);
                   All the remaining matrices should be zero

> for i from 1 to 3 do
>   for j from 1 to 3 do
>   A := matrix(3,3,0);
>   A[i,j]:=1;
>   if i=j then A[3,3]:=-A[i,i] fi;
>   xA := so7Lb(X,sl3(A));
>   print(map(simplify,J2stP(xA,1)));
>   od;
> od;
bytes used=11112456, alloc=2227816, time=17.43
bytes used=12112684, alloc=2227816, time=19.23
bytes used=13115316, alloc=2227816, time=21.65
bytes used=14125600, alloc=2293340, time=26.52
                       [0    0    0    0    0    0    0]
                       [                               ]
                       [0    0    0    0    0    0    0]
                       [                               ]
                       [0    0    0    0    0    0    0]
                       [                               ]
```

```
                      [0    0    0    0    0    0    0]
                      [                              ]
                      [0    0    0    0    0    0    0]
                      [                              ]
                      [0    0    0    0    0    0    0]
                      [                              ]
                      [0    0    0    0    0    0    0]

bytes used=15125800, alloc=2293340, time=27.92
bytes used=16126132, alloc=2293340, time=29.34
                      [0    0    0    0    0    0    0]
                      [                              ]
                      [0    0    0    0    0    0    0]
                      [                              ]
                      [0    0    0    0    0    0    0]
                      [                              ]
                      [0    0    0    0    0    0    0]
                      [                              ]
                      [0    0    0    0    0    0    0]
                      [                              ]
                      [0    0    0    0    0    0    0]
                      [                              ]
                      [0    0    0    0    0    0    0]

bytes used=17126340, alloc=2358864, time=30.77
bytes used=18126556, alloc=2358864, time=32.20
                      [0    0    0    0    0    0    0]
                      [                              ]
                      [0    0    0    0    0    0    0]
                      [                              ]
                      [0    0    0    0    0    0    0]
                      [                              ]
                      [0    0    0    0    0    0    0]
                      [                              ]
                      [0    0    0    0    0    0    0]
                      [                              ]
                      [0    0    0    0    0    0    0]
                      [                              ]
                      [0    0    0    0    0    0    0]

bytes used=19126764, alloc=2358864, time=33.70
bytes used=20127012, alloc=2358864, time=35.44
bytes used=21127412, alloc=2358864, time=37.12
bytes used=22127876, alloc=2358864, time=38.39
bytes used=23128584, alloc=2358864, time=39.65
```

```
bytes used=24128804, alloc=2358864, time=40.88
bytes used=25129248, alloc=2358864, time=42.58
bytes used=26129648, alloc=2358864, time=43.89
bytes used=27130100, alloc=2358864, time=45.17
bytes used=28279020, alloc=2358864, time=47.03
                       [0    0    0    0    0    0    0]
                       [                               ]
                       [0    0    0    0    0    0    0]
                       [                               ]
                       [0    0    0    0    0    0    0]
                       [                               ]
                       [0    0    0    0    0    0    0]
                       [                               ]
                       [0    0    0    0    0    0    0]
                       [                               ]
                       [0    0    0    0    0    0    0]
                       [                               ]
                       [0    0    0    0    0    0    0]

bytes used=29279808, alloc=2555436, time=48.66
bytes used=30280036, alloc=2555436, time=50.20
bytes used=31455428, alloc=2555436, time=53.62
                       [0    0    0    0    0    0    0]
                       [                               ]
                       [0    0    0    0    0    0    0]
                       [                               ]
                       [0    0    0    0    0    0    0]
                       [                               ]
                       [0    0    0    0    0    0    0]
                       [                               ]
                       [0    0    0    0    0    0    0]
                       [                               ]
                       [0    0    0    0    0    0    0]
                       [                               ]
                       [0    0    0    0    0    0    0]

bytes used=32455716, alloc=2620960, time=55.37
bytes used=33455932, alloc=2620960, time=56.90
                       [0    0    0    0    0    0    0]
                       [                               ]
                       [0    0    0    0    0    0    0]
                       [                               ]
                       [0    0    0    0    0    0    0]
                       [                               ]
                       [0    0    0    0    0    0    0]
```

```
                              [                             ]
                              [0    0    0    0    0    0    0]
                              [                             ]
                              [0    0    0    0    0    0    0]
                              [                             ]
                              [0    0    0    0    0    0    0]

bytes used=34456196, alloc=2620960, time=58.33
bytes used=35456540, alloc=2620960, time=59.89
                              [0    0    0    0    0    0    0]
                              [                             ]
                              [0    0    0    0    0    0    0]
                              [                             ]
                              [0    0    0    0    0    0    0]
                              [                             ]
                              [0    0    0    0    0    0    0]
                              [                             ]
                              [0    0    0    0    0    0    0]
                              [                             ]
                              [0    0    0    0    0    0    0]
                              [                             ]
                              [0    0    0    0    0    0    0]

bytes used=36456760, alloc=2620960, time=61.56
bytes used=37457132, alloc=2620960, time=63.32
                              [0    0    0    0    0    0    0]
                              [                             ]
                              [0    0    0    0    0    0    0]
                              [                             ]
                              [0    0    0    0    0    0    0]
                              [                             ]
                              [0    0    0    0    0    0    0]
                              [                             ]
                              [0    0    0    0    0    0    0]
                              [                             ]
                              [0    0    0    0    0    0    0]
                              [                             ]
                              [0    0    0    0    0    0    0]

                    Warning: matrix passed to sl3 is not trace-free

bytes used=38457428, alloc=2620960, time=64.85
bytes used=39457736, alloc=2620960, time=66.36
bytes used=40767288, alloc=2817532, time=69.89
                 [                3  1/2                 2         ]
```

```
              [               t  2                   t  s       ]
              [0 , 0 , 0 , - --------- , 0 , - 2/3 --------- , 0]
              [               2      2              2      2    ]
              [              s  + 9 t              s  + 9 t     ]
              [                                                 ]
              [                                           2     ]
              [                                          t  s   ]
              [0 ,   0 ,   0 ,   0 ,   0 ,   0 ,   2/3 ---------]
              [                                         2      2]
              [                                        s  + 9 t ]
              [                                                 ]
              [0 ,     0 ,     0 ,     0 ,     0 ,     0 ,     0]
              [                                                 ]
              [                 2    1/2                 3  1/2 ]
              [                t  s 2                   t  2    ]
              [0 ,  0 ,  - 1/3 --------- , 0 , 0 , 0 , ---------]
              [                 2      2                2      2]
              [                s  + 9 t                s  + 9 t ]
              [                                                 ]
              [            3                2    1/2            ]
              [           t                t  s 2               ]
              [0 ,  -2 --------- , 0 , 1/3 --------- , 0 , 0 , 0]
              [         2      2            2      2            ]
              [        s  + 9 t            s  + 9 t             ]
              [                                                 ]
              [                    3                            ]
              [                   t                             ]
              [0 ,    0 ,    2 --------- ,   0 ,   0 ,   0 ,   0]
              [                 2      2                        ]
              [                s  + 9 t                         ]
              [                                                 ]
              [0 ,     0 ,     0 ,     0 ,     0 ,     0 ,     0]

> for i from 1 to 3 do
>   v:=[0,0,0];
>   v[i]:=1;
>   xA := so7Lb(X,V10(v));
>   print(map(simplify,J2stP(xA,1)));
>   xA := so7Lb(X,V01(v));
>   print(map(simplify,J2stP(xA,1)));
> od;
                                 v := [0, 0, 0]

                                   v[1] := 1
```

```
                          [0    0    0    0    0    0    0]
                          [                               ]
                          [0    0    0    0    0    0    0]
                          [                               ]
                          [0    0    0    0    0    0    0]
                          [                               ]
                    xA := [0    0    0    0    0    0    0]
                          [                               ]
                          [0    0    0    0    0    0    0]
                          [                               ]
                          [0    0    0    0    0    0    0]
                          [                               ]
                          [0    0    0    0    0    0    0]

bytes used=41767624, alloc=2817532, time=71.76
bytes used=42767836, alloc=2817532, time=73.28
                       [0    0    0    0    0    0    0]
                       [                               ]
                       [0    0    0    0    0    0    0]
                       [                               ]
                       [0    0    0    0    0    0    0]
                       [                               ]
                       [0    0    0    0    0    0    0]
                       [                               ]
                       [0    0    0    0    0    0    0]
                       [                               ]
                       [0    0    0    0    0    0    0]
                       [                               ]
                       [0    0    0    0    0    0    0]

                   [   1/2                            1/2        ]
                   [t 2    ,  0 ,  0 ,  0 ,  - 1/2 s 2    , 0 , 0]
                   [                                             ]
                   [               1/2                           ]
                   [0 ,   - 1/2 t 2    ,   0 ,   0 ,  0 ,  0 ,  0]
                   [                                             ]
                   [                 1/2                      1/2]
                   [0 , 0 , - 1/2 t 2    , 0 , 0 , 0 , 1/2 s 2   ]
                   [                                             ]
             xA := [0 ,     -s ,    0 ,    0 ,    0 ,    0 ,    0]
                   [                                             ]
                   [                               1/2           ]
                   [0 ,   0 ,   0 ,   0 ,   1/2 t 2    ,   0 ,  0]
                   [                                             ]
                   [                                     1/2     ]
```

```
                    [0 ,   0 ,   0 ,   s ,   0 ,   1/2 t 2    ,  0]
                    [                                             ]
                    [                                          1/2]
                    [0 ,    0 ,    0 ,   0 ,   0 ,   0 ,   -t 2   ]

bytes used=43768040, alloc=2817532, time=74.76
bytes used=44768288, alloc=2817532, time=76.41
bytes used=45768500, alloc=2817532, time=78.37
bytes used=46768864, alloc=2817532, time=79.58
bytes used=47769256, alloc=2817532, time=80.83
bytes used=48769516, alloc=2817532, time=82.05
bytes used=49769924, alloc=2817532, time=83.30
bytes used=50770316, alloc=2817532, time=84.65
bytes used=51770660, alloc=2817532, time=85.87
bytes used=52771260, alloc=2817532, time=87.12
bytes used=53771468, alloc=2817532, time=88.33
bytes used=54771720, alloc=2817532, time=89.58
bytes used=55771988, alloc=2817532, time=90.84
bytes used=56772272, alloc=2817532, time=93.18
                        [0    0    0    0    0    0    0]
                        [                               ]
                        [0    0    0    0    0    0    0]
                        [                               ]
                        [0    0    0    0    0    0    0]
                        [                               ]
                        [0    0    0    0    0    0    0]
                        [                               ]
                        [0    0    0    0    0    0    0]
                        [                               ]
                        [0    0    0    0    0    0    0]
                        [                               ]
                        [0    0    0    0    0    0    0]

                                  v := [0, 0, 0]

                                    v[2] := 1

         [                                                    1/2          ]
         [0         0              0          s      0    -t 2         0   ]
         [                                                                 ]
         [                                                              1/2]
         [0         0              0          0      0       0       t 2   ]
         [                                                                 ]
         [0         0              0          0      0       0         0   ]
         [                                                                 ]
```

```
   xA := [0         0             2 t         0      0       0         -s  ]
         [                                                                 ]
         [              1/2                                                ]
         [0    - 1/2 s 2           0         -2 t    0       0         0   ]
         [                                                                 ]
         [                            1/2                                  ]
         [0         0          1/2 s 2        0      0       0         0   ]
         [                                                                 ]
         [0         0              0          0      0       0         0   ]

bytes used=57772644, alloc=2817532, time=95.06
bytes used=58773524, alloc=2817532, time=96.85
bytes used=60089240, alloc=2817532, time=101.53
                       [0    0    0    0    0    0    0]
                       [                               ]
                       [0    0    0    0    0    0    0]
                       [                               ]
                       [0    0    0    0    0    0    0]
                       [                               ]
                       [0    0    0    0    0    0    0]
                       [                               ]
                       [0    0    0    0    0    0    0]
                       [                               ]
                       [0    0    0    0    0    0    0]
                       [                               ]
                       [0    0    0    0    0    0    0]

                [            1/2                                    ]
                [0    3/2 t 2       0    0    0    0         0      ]
                [                                                   ]
                [0        0         0    0    0    0         0      ]
                [                                                   ]
                [0        0         0    0    0    0         0      ]
                [                                                   ]
          xA := [0        0         0    0    0    0         0      ]
                [                                                   ]
                [0        0         0    0    0    0         0      ]
                [                                                   ]
                [                                                1/2]
                [0        0         0    0    0    0    - 3/2 t 2   ]
                [                                                   ]
                [0        0         0    0    0    0         0      ]

bytes used=61089588, alloc=2817532, time=103.42
bytes used=62089912, alloc=2817532, time=104.80
```

```
bytes used=63429108, alloc=2817532, time=108.52
                       [0    0    0    0    0    0    0]
                       [                               ]
                       [0    0    0    0    0    0    0]
                       [                               ]
                       [0    0    0    0    0    0    0]
                       [                               ]
                       [0    0    0    0    0    0    0]
                       [                               ]
                       [0    0    0    0    0    0    0]
                       [                               ]
                       [0    0    0    0    0    0    0]
                       [                               ]
                       [0    0    0    0    0    0    0]

                                 v := [0, 0, 0]

                                   v[3] := 1

                  [                             1/2               ]
                  [0     0      0     0     -t 2       0      0   ]
                  [                                               ]
                  [0     0      0     0        0       0      0   ]
                  [                                               ]
                  [                                            1/2]
                  [0     0      0     0        0       0    t 2   ]
            xA := [                                               ]
                  [0    -2 t    0     0        0       0      0   ]
                  [                                               ]
                  [0     0      0     0        0       0      0   ]
                  [                                               ]
                  [0     0      0    2 t       0       0      0   ]
                  [                                               ]
                  [0     0      0     0        0       0      0   ]

bytes used=64429320, alloc=2817532, time=110.34
bytes used=65429564, alloc=2817532, time=111.88
bytes used=66430064, alloc=2817532, time=113.44
bytes used=67430428, alloc=2817532, time=114.67
bytes used=68430832, alloc=2817532, time=115.98
bytes used=69431136, alloc=2817532, time=117.24
bytes used=70460092, alloc=2817532, time=118.77
                       [0    0    0    0    0    0    0]
                       [                               ]
                       [0    0    0    0    0    0    0]
```

```
                       [                               ]
                       [0    0    0    0    0    0    0]
                       [                               ]
                       [0    0    0    0    0    0    0]
                       [                               ]
                       [0    0    0    0    0    0    0]
                       [                               ]
                       [0    0    0    0    0    0    0]
                       [                               ]
                       [0    0    0    0    0    0    0]

                [                 1/2                               ]
                [0    0    3/2 t 2       0    0    0         0      ]
                [                                                   ]
                [0    0        0         0    0    0         0      ]
                [                                                   ]
                [0    0        0         0    0    0         0      ]
                [                                                   ]
          xA := [0    0        0         0    0    0         0      ]
                [                                                   ]
                [                                                1/2]
                [0    0        0         0    0    0    - 3/2 t 2   ]
                [                                                   ]
                [0    0        0         0    0    0         0      ]
                [                                                   ]
                [0    0        0         0    0    0         0      ]

bytes used=71460332, alloc=2817532, time=120.37
bytes used=72460560, alloc=2817532, time=121.95
                       [0    0    0    0    0    0    0]
                       [                               ]
                       [0    0    0    0    0    0    0]
                       [                               ]
                       [0    0    0    0    0    0    0]
                       [                               ]
                       [0    0    0    0    0    0    0]
                       [                               ]
                       [0    0    0    0    0    0    0]
                       [                               ]
                       [0    0    0    0    0    0    0]
                       [                               ]
                       [0    0    0    0    0    0    0]

> quit;
bytes used=72775452, alloc=2817532, time=122.39
```

List of Code Chunks

Index

References

Knuth, D. E. (1992). *Literate Programming*, CSLI Lecture Notes Number 27, Stanford University Center for the Study of Language and Information, Stanford, CA, USA. 3

Kobak, P. Z. and Swann, A. F. (1993). Quaternionic geometry of a nilpotent variety, *Math. Ann.* **297**: 747–764. 2

Kobak, P. Z. and Swann, A. F. (1998). HyperKähler potentials in cohomogeneity two, *preprint 98/33*, Department of Mathematical Sciences, University of Bath. 2, 2, 2, 15

Kobak, P. Z. and Swann, A. F. (1999). The hyperKähler geometry associated to Wolf spaces, *preprint 99/14*, Department of Mathematical Sciences, University of Bath. 2

Kronheimer, P. B. (1990). Instantons and the geometry of the nilpotent variety, *J. Differential Geom.* **32**: 473–490. 2

Swann, A. F. (1991). HyperKähler and quaternionic Kähler geometry, *Math. Ann.* **289**: 421–450. 2