```
(* ============================================================================== *)
(*                            Dirac-Lord-Nash_4+4.wl                              *)
(* ======================  a few references:   ================================= *)
(*JOURNAL OF NATHENATICAL PHYSICS, VOLUME 4, NUMBER 7, JULY 1963*)
(*"A Remarkable Representation of the 3 + 2 de Sitter Group"*)
(*P. A. M. DIrac*)
(* ============================================================================== *)
(*Proc. Camb. Phil. Soc. (1968), 64, 765*)
(*"The Dirac spinor in six dimensions"*)
(*E. A. LORD*)
(*Department of Mathematics, King's College, University of London*)
(* ============================================================================== *)
(*J. Math. Phys. 25 (2), February 1984*)
(*"Identities satisfied by the generators of the Dirac algebra"*)
(*Patrick L. Nash*)
(* ============================================================================== *)
(*IL NUOVO CIMENTO, VoL. 105 B, N. 1, Gennaio 1990*)
(*"On the Structure of the Split Octonion Algebra"*)
(*P. L. NASH*)
(*University of Texas at San Antonio, TX 78285-0663*)
(* ============================================================================== *)
(*JOURNAL OF MATHEMATICAL PHYSICS 51, 042501 (2010)*)
(*"Second gravity"*)
(*Patrick L. Nash*)
(* ============================================================================== *)
(*                                                                          *)
(*  Clifford Algebra Cl(4,4) and Spin(4,4) Representations                  *)
(*  for Split Octonions and Cartan's Triality                               *)
(*                                                                          *)
(*  This package provides:                                                  *)
(*    1. Real 16x16 matrix representation of Cl(4,4) with generators t^A    *)
(*    2. Two real 8x8 matrix representations of Spin(4,4)                   *)
(*    3. Proof of anti-commutation relations {t^A, t^B} = 2 eta^{AB} I_16   *)
(*    4. Proof of commutation relations for spin generators S^{AB}          *)
(*                                                                          *)
(*  Mathematical Background:                                                *)
(*    - Split octonions Os: 8D non-associative algebra over R               *)
(*    - Signature (4,4): <x,x> = x0^2+x1^2+x2^2+x3^2-x4^2-x5^2-x6^2-x7^2    *)
(*    - Cartan's triality: V, S1, S2 are equivalent 8D representations      *)
(*                                                                          *)
(*  Usage: Get["DiracLordNash4+4.wl"]                                       *)
(*                                                                          *)
(* ============================================================================== *)
(* ================================================================================ *)
(* Patrick L. Nash, Ph.D.     (c) 2022, under GPL ; do not remove this notice   *)
(* Professor, UTSA Physics and Astronomy, Retired (UTSA)                        *)
```

```
(* Patrick299Nash  at    gmail   ...                                        *)
(* Enhanced Version 2 - Fixed HTML entity handling and partial derivatives      *)
(* blame: PLN and friends (Claude Opus 4.5 and Manus-Lite)                          *)
(* ============================================================================ *)

BeginPackage["DiracLordNash44`"];

(* ========================================================================== *)
(*                          TABLE OF CONTENTS                                  *)
(* ========================================================================== *)
(*                                                                             *)
(*  SECTION 1: Basic Definitions and Identity Matrices                         *)
(*  SECTION 2: Metric Tensors                                                  *)
(*  SECTION 3: Pauli-like 2x2 Building Blocks                                  *)
(*  SECTION 4: Self-Dual and Anti-Self-Dual 4x4 Matrices                        *)
(*  SECTION 5: 8x8 Clifford Algebra Generators Q[A] for spinor space        *)
(*  SECTION 6: Conjugate Q-bar generators                                     *)
(*  SECTION 7: 16x16 Clifford Algebra Generators T16^A[n]                      *)
(*  SECTION 8: Chirality and Volume Elements                                   *)
(*  SECTION 9: Spin(4,4) Generators S^{AB} (8x8 reducible representations)     *)
(*  SECTION 10: Verification of Anti-Commutation Relations                     *)
(*  SECTION 11: Verification of Commutation Relations                          *)
(*  SECTION 12: Helper Functions for Lagrangian Construction                   *)
(*  SECTION 13: Unit Spinor, F-matrices, Projections, Fundamental Identity    *)
(*  SECTION 14: Complete 256-Element Basis via Pauli Kronecker Products       *)
(*                                                                             *)
(* ========================================================================== *)

(* Public symbols *)
ID2::usage = "ID2 is the 2x2 identity matrix.";
ID4::usage = "ID4 is the 4x4 identity matrix.";
ID8::usage = "ID8 is the 8x8 identity matrix.";
ID16::usage = "ID16 is the 16x16 identity matrix.";

eta2244::usage = "eta2244 is the 4x4 metric with signature (2,2): diag(-1,1,-1,1).";
etaAB::usage = "etaAB is the 8x8 metric with signature (4,4): diag(1,1,1,1,-1,-1,-1,-1).

σ22::usage = "σ22 is a list of four real 2x2 matrices forming a basis.";
σBar22::usage = "σBar22 is the conjugate basis with -I2 as first element.";

s4by4::usage = "s4by4[h] gives the h-th self-dual antisymmetric 4x4 matrix (h=1,2,3).";
t4by4::usage = "t4by4[h] gives the h-th anti-self-dual antisymmetric 4x4 matrix (h=1,2,3

allS4by4::usage = "gives all s4by4 self-dual antisymmetric 4x4 matrix (h=1,2,3).";
allT4by4::usage = "gives all s4by4 anti-dual antisymmetric 4x4 matrix (h=1,2,3).";

(* OverBar[allQ] usage is documented via allQBar below *)
```

```
allQ::usage = "allQ";
allQBar::usage = "allQBar";
Q::usage = "Q[A] gives the A-th 8x8 Clifford generator (A=0,...,7). Q[0]=ID8.";
QBar::usage = "QBar[A] (or OverBar[Q][A]) gives the conjugate of Q[A] via QBar[A] = -eta

T16A::usage = "T16A[n] gives the n-th 16x16 Clifford algebra generator (n=0,...,8).";

σ8::usage = "σ8 is the 8x8 chirality matrix: Q[1].Q[2].Q[3].";
σ16::usage = "σ16 is the 16x16 chirality matrix.";
Omega8::usage = "Omega8 is the 8x8 volume element: σ8.Q[7].";
Omega16::usage = "Omega16 is the 16x16 complex structure matrix.";

(*SAB8::usage = "SAB8[A,B] incorrectly gives the (A,B) Spin(4,4) generator as an 8x8 mat
SAB16::usage = "SAB16[A,B] gives the (A,B) Spin(4,4) generator as a 16x16 matrix.";
SAB::usage = "gives ALL Spin(4,4) generator as a 16x16 matrix.";

SAB1::usage = "SAB1 returns table (A,B) Spin(4,4) generator as an 8x8 matrix (acts on S1
SAB2::usage = "SAB2 returns table (A,B) Spin(4,4) generator as an 8x8 matrix (acts on S2)

SpinorMetric8::usage = "SpinorMetric8 is the 8x8 spinor metric C = {{0,I4},{I4,0}}.";
SpinorMetric16::usage = "SpinorMetric16 is the 16x16 spinor metric.";

verifyAntiCommutation::usage = "verifyAntiCommutation[] returns True if all anti-commutat
verifyCommutation::usage = "verifyCommutation[] returns True if all spin generator commut

(* Section 13: Unit Spinor and Lagrangian Construction *)
unit::usage = "unit is the unit type-1 spinor, an eigenspinor of σ8 with eigenvalue +1.";
FAa::usage = "FAa is the 8x8 matrix F_A^a = η_{AA} * (τ[A] . unit)^T for Lagrangian cons
FaA::usage = "FaA is the list of row vectors F_a^A = unit^T . σ8 . τ¯[A] for A=0,...,7.";
FForthogonality::usage = "FForthogonality is the 8x8 matrix FaA . FAa, which should equal
splitOctonionMult::usage = "splitOctonionMult[A,B,C] gives the split octonion structure c
(*EA::usage = "mult tab entries";*)
eA::usage = "mult tab entries";
times::usage = "mult tab entries";
splitOctonionMultTable::usage = "splitOctonionMultTable gives the split octonion multipli
realProjection8::usage = "realProjection8 is the 8x8 real projection matrix: KroneckerPro
realProjection16::usage = "realProjection16 is the 16x16 real projection: {{realProjectic
imaginaryPart8::usage = "imaginaryPart8[ψ] returns the imaginary (non-real) part of 8-sp
imaginaryPart16::usage = "imaginaryPart16[Ψ] returns the imaginary part of 16-spinor Ψ.";
FundamentalIdentity8by8::usage = "FundamentalIdentity8by8[a] verifies Tr[a]*I8 = Σ η[A,A
testFundamentalIdentity::usage = "testFundamentalIdentity[matrix] tests fundamental ident
fundamentalIdentityTest1::usage = "fundamentalIdentityTest1 is True if fundamental identi
fundamentalIdentityTest2::usage = "fundamentalIdentityTest2 is True if fundamental identi
fundamentalIdentityTest3::usage = "fundamentalIdentityTest3 is True if fundamental identi

(* Section 14: 256-Element Basis *)
```

```
pauli::usage = "pauli[k] returns the k-th Pauli matrix: pauli[0]=I2, pauli[1,2,3]=PauliM
pauliReal::usage = "pauliReal[k] returns the k-th REAL Pauli basis: pauliReal[2]=I*PauliI
Basis16::usage = "Basis16[a,b,c,d] returns the 16x16 matrix σ_a⊗σ_b⊗σ_c⊗σ_d (a,b,c,d∈{0,
Basis16Real::usage = "Basis16Real[a,b,c,d] returns the REAL 16x16 matrix using pauliReal
Basis16Index::usage = "Basis16Index[a,b,c,d] returns linear index n = 64a+16b+4c+d ∈ {0,
Basis16FromIndex::usage = "Basis16FromIndex[n] returns {a,b,c,d} from linear index n.";
Basis16ByIndex::usage = "Basis16ByIndex[n] returns the n-th basis matrix (n∈{0,...,255})
Basis16Label::usage = "Basis16Label[a,b,c,d] returns string label like 'σ1 ⊗ I ⊗ σ3 ⊗ σ2
ViewBasis16::usage = "ViewBasis16[a,b,c,d] displays basis matrix with label and index.";
ViewBasis16ByIndex::usage = "ViewBasis16ByIndex[n] displays the n-th basis matrix.";
GenerateAllBasis16::usage = "GenerateAllBasis16[] returns list of all 256 {index,label,ma
AllBasis16::usage = "AllBasis16 is a cached list of all 256 basis matrices.";
AllBasis16Real::usage = "AllBasis16Real is a cached list of all 256 REAL basis matrices."
Basis16IndexTable::usage = "Basis16IndexTable[] displays table of all 256 indices and lab
ExpandInBasis16::usage = "ExpandInBasis16[M] returns 256 coefficients of M in the Pauli b
NonZeroComponents16::usage = "NonZeroComponents16[M] returns non-zero basis components of
VerifyBasis16Orthogonality::usage = "VerifyBasis16Orthogonality[] returns True if basis i
(*X::usage = "default Minkowski coored";*)
epsilon3::usage = "Levi-Civita symbol for 3 indices";
epsilon4::usage = "Levi-Civita symbol for 4 indices";

Begin["`Private`"];

(* ========================================================================= *)
(*  SECTION 1: Basic Definitions and Identity Matrices                      *)
(* ========================================================================= *)
(*X = {x0, x1, x2, x3, x4, x5, x6, x7};
Protect[X];
Protect[x0, x1, x2, x3, x4, x5, x6, x7];*)

ID2 = IdentityMatrix[2];
ID4 = IdentityMatrix[4];
ID8 = IdentityMatrix[8];
ID16 = IdentityMatrix[16];

(* Zero matrices for convenience *)
Zero4 = Array[0 &, {4, 4}];
Zero8 = Array[0 &, {8, 8}];

(* ========================================================================= *)
(*  SECTION 2: Metric Tensors                                               *)
(* ========================================================================= *)

(* 4x4 metric with signature (2,2) for building blocks *)
eta2244 = DiagonalMatrix[{-1, 1, -1, 1}];

(* 8x8 metric with signature (4,4) for split octonions *)
```

```
(* Indices: 0,1,2,3 are timelike (+1), 4,5,6,7 are spacelike (-1) *)
etaAB = ArrayFlatten[{{ID4, Zero4}, {Zero4, -ID4}}];

(* Levi-Civita symbol for 4 indices *)
epsilon4 = Array[Signature[{##}] &, {4, 4, 4, 4}];
epsilon3 = Array[Signature[{##}]&,{3,3,3}]
(* ========================================================================== *)
(*  SECTION 3: Pauli-like 2x2 Building Blocks                                *)
(* ========================================================================== *)

(* Real 2x2 matrices forming a Clifford algebra basis *)
(* σ22 = {I2, σ_1, i*σ_2, σ_3} where i*σ_2 is real *)
σ22 = {
    IdentityMatrix[2],           (* {{1,0},{0,1}} *)
    PauliMatrix[1],              (* {{0,1},{1,0}} *)
    I * PauliMatrix[2],          (* {{0,1},{-1,0}} - real! *)
    PauliMatrix[3]               (* {{1,0},{0,-1}} *)
};

(* Conjugate basis with opposite first element *)
σBar22 = {
    -IdentityMatrix[2],          (* {{-1,0},{0,-1}} *)
    PauliMatrix[1],              (* {{0,1},{1,0}} *)
    I * PauliMatrix[2],          (* {{0,1},{-1,0}} *)
    PauliMatrix[3]               (* {{1,0},{0,-1}} *)
};

(* ========================================================================== *)
(*  SECTION 4: Self-Dual and Anti-Self-Dual 4x4 Matrices                     *)
(* ========================================================================== *)

(* Functions to build 4x4 blocks from 2x2 matrices via Kronecker products *)
yyy[j_] := KroneckerProduct[σ22⟦j⟧, σ22⟦2⟧];
xxx[j_] := ArrayFlatten[{{σ22⟦j⟧, 0}, {0, σBar22⟦j⟧}}];

(* Self-dual antisymmetric 4x4 matrices (h = 1,2,3) *)
(* These satisfy: (1/2)*epsilon[p,q,j1,j2]*s4by4[h]⟦j1,j2⟧ = s4by4[h]⟦p,q⟧ *)
```

```
(* Anti-self-dual antisymmetric 4x4 matrices (h = 1,2,3) *)
(* These satisfy: (1/2)*epsilon[p,q,j1,j2]*t4by4[h]⟦j1,j2⟧ = -t4by4[h]⟦p,q⟧ *)

Qa1234[h_, p_, q_] := Signature[{h, p, q, 4}];
Qb1234[h_, p_, q_] := ID4⟦p, 4⟧*ID4⟦q, h⟧ - ID4⟦p, h⟧*ID4⟦q, 4⟧;
SelfDualAntiSymmetric[h_, p_, q_] := Qa1234[h, p, q] - Qb1234[h, p, q] ;
AntiSelfDualAntiSymmetric[h_, p_, q_] := (Qa1234[h, p, q] + Qb1234[h, p, q] );
```

```
allS4by4=Table[s4by4[h] = Table[Table[SelfDualAntiSymmetric[h, p, q], {q, 4}], {p, 4}],
allT4by4=Table[t4by4[h] = Table[Table[AntiSelfDualAntiSymmetric[h, p, q], {q, 4}], {p, 4

(* ========================================================================== *)
(*  SECTION 5: 8x8 Clifford Algebra Generators Q[A]                          *)
(* ========================================================================== *)

(* Q[0] = identity (required for completeness) *)
Q[0] = ID8;
OverBar[Q][0] = ID8;
Table[Q[7 - h] = ArrayFlatten[{{0, t4by4[h]}, {-t4by4[h], 0}}], {h, 1, 3} ];
Table[Q[h] = ArrayFlatten[{{0, s4by4[h]}, {s4by4[h], 0}}], {h, 1, 3} ];

(* Q[1], Q[2], Q[3]: Built from self-dual matrices *)
(* These are symmetric: Q[h] = Transpose[Q[h]] for h = 1,2,3 *)
(*Do[
    Q[h] = ArrayFlatten[{{0, s4by4[h]}, {s4by4[h], 0}}],
    {h, 1, 3}
];
*)
(* Q[4], Q[5], Q[6]: Built from anti-self-dual matrices *)
(* These are antisymmetric: Q[h] = -Transpose[Q[h]] for h = 4,5,6 *)
(*Do[
    Q[7 - h] = ArrayFlatten[{{0, t4by4[h]}, {-t4by4[h], 0}}],
    {h, 1, 3}
];*)

(* Q[7]: The chirality-related generator, defined as product of others *)
Q[7] = Q[1] . Q[2] . Q[3] . Q[4] . Q[5] . Q[6];
σ8 = Q[1] . Q[2] . Q[3];
(* ========================================================================== *)
(*  SECTION 6: Conjugate Q-bar Generators                                    *)
(* ========================================================================== *)

(* The conjugate generators satisfy: OverBar[Q][A] = -eta[A,A] * Transpose[Q[A]] *)
(* For A = 1,2,3: eta[A,A] = +1, so OverBar[Q][A] = -Transpose[Q[A]] *)
(* For A = 4,5,6,7: eta[A,A] = -1, so OverBar[Q][A] = Transpose[Q[A]] *)
OverBar[allQ]=Table[OverBar[Q][A1] = σ8 . Transpose[σ8 . Q[A1]],{A1, 1, 7}];
PrependTo[OverBar[allQ],OverBar[Q][0]];
allQ   = Table[Q[A1] ,{A1, 0, 7}];
allQBar=Table[OverBar[Q][A1],{A1, 0, 7}];
```

```
(* ============================================================================ *)
(*  SECTION 7: 16x16 Clifford Algebra Generators T16^A[n]                     *)
(* ============================================================================ *)

(* The 16x16 generators act on the full spinor space S1 ⊕ S2 *)
(* Construction: T16^A[n] = {{0, OverBar[Q][n]}, {Q[n], 0}} *)
allT16A=Table[T16A[A1] = ArrayFlatten[{{0, OverBar[Q][A1]}, {Q[A1], 0}}],{A1, 0, 7}
];

(* T16^A[8]: The 16D chirality element (product of all generators) *)
T16A[8] = T16A[0] . T16A[1] . T16A[2] . T16A[3] . T16A[4] . T16A[5] . T16A[6] . T16A[7]
AppendTo[allT16A,T16A[8]];
(* ============================================================================ *)
(*  SECTION 8: Chirality and Volume Elements                                   *)
(* ============================================================================ *)

(* 8x8 chirality matrix: σ = Q[1].Q[2].Q[3] *)
(* This has eigenvalues +1 and -1, projecting onto type-1 and type-2 spinor spaces *)
(*σ8 = Q[1] . Q[2] . Q[3];*)

(* Alternative representation: σ8 = Q[4].Q[5].Q[6].Q[7] *)
(* Verification: σ8 == Q[4].Q[5].Q[6].Q[7] should be True *)

(* 16x16 chirality matrix *)
σ16 = T16A[0] . T16A[1] . T16A[2] . T16A[3];

σ16 . T16A[♯] == -Transpose[σ16 . T16A[♯]] & /@ Range[0, 7]

(* Relation: σ16 == ArrayFlatten[{{-σ8, 0}, {0, σ8}}] *)

(* 8x8 volume element (complex structure) *)
Omega8 = σ8 . Q[7];

(* 16x16 complex structure *)
Omega16 = T16A[0] . T16A[4] . σ16;

(* ============================================================================ *)
(*  SECTION 9: Spin(4,4) Generators S^{AB}                                     *)
(* ============================================================================ *)

(* The spin generators are defined as commutators: S^{AB} = (1/4)(t^A.t^B - t^B.t^A) *)
(* These form the Lie algebra so(4,4) *)

(* WTF:   8x8 spin generators (act on S1 or S2 individually) *)
(*SAB8[A_, B_] := (1/4) * (Q[A] . Q[B] - Q[B] . Q[A]);*)

```

```
SAB1=Table[1/4 ( OverBar[Q][A1] . Q[B1]-OverBar[Q][B1] . Q[A1]),{A1,0, 7},{B1,0, 7}]
SAB2=Table[1/4 ( Q[A1] . OverBar[Q][B1]-Q[B1] . OverBar[Q][A1]),{A1,0, 7},{B1,0, 7}]
```

```
(* 16x16 spin generators (act on S1 ⊕ S2) *)
SAB = Table[1/4 ((T16^A)[A1] . (T16^A)[B1] - (T16^A)[B1] . (T16^A)[A1]), {A1, 0, 7}, {B
SAB16[A_, B_] :=SAB⟦A,B⟧;   (*(1/4) * (T16A[A] . T16A[B] - T16A[B] . T16A[A]);*)
```

```
(* Note: S^{AB} = -S^{BA} (antisymmetric) *)
(* Note: S^{AA} = 0 for all A *)

(* ============================================================================ *)
(*  SECTION 10: Verification of Anti-Commutation Relations                   *)
(* ============================================================================ *)

(* The Clifford algebra Cl(4,4) is defined by: {t^A, t^B} = 2*eta^{AB}*I *)
(* That is: t^A.t^B + t^B.t^A = 2*etaAB⟦A+1,B+1⟧*I *)

(* Verification function for 8x8 generators *)
verifyAntiCommutation8[] := Module[{result = True, antiComm},
    Do[
        antiComm = Q[A] . OverBar[Q][B] + Q[B] . OverBar[Q][A]//FullSimplify;
        If[antiComm ≠ 2 * etaAB⟦A + 1, B + 1⟧ * ID8,
            result = False;
            Print["Anti-commutation 8 fails for A=", A, ", B=", B, ", ==", antiComm];
        ],
        {A, 0, 7}, {B, 0, 7}
    ];
    result
];

(* Verification function for 16x16 generators *)
verifyAntiCommutation16[] := Module[{result = True, antiComm},
    Do[
        antiComm = T16A[A] . T16A[B] + T16A[B] . T16A[A];
        If[antiComm ≠ 2 * etaAB⟦A + 1, B + 1⟧ * ID16,
            result = False;
            Print["Anti-commutation 16 fails for A=", A, ", B=", B];
        ],
        {A, 0, 7}, {B, 0, 7}
```

```
Print["Dirac-Lord-Nash_4+4 loaded successfully!  BUT, WARNING:  DO NOT USE IF YOU WANT A
  result = Transpose[unit] . OverBar[Q][A] . Q[B] . Q[C] . unit;
  (* Apply metric factors for proper index placement *)
  etaAB⟦A + 1, A + 1⟧ * etaAB⟦B + 1, B + 1⟧ * etaAB⟦C + 1, C + 1⟧ * result
];
```