```
(*
equations3and4and28.nb

 produced by D.F.Gochberg for paper:
  D.F.Gochberg, M.D.Does, Z.Zu, C.L.Lankford. Towards
  an analytic solution for pulsed CEST.NMR in Biomedicine 2017

  You are free to use this code for non-commercial purposes,
but please cite the above manuscript if you use the code,or parts thereof,
to help produce a manuscript or presentation figure,as appropriate.Thanks.

  Based on: analytic_pulsed_CEST6.nb (Dan's reminder to himself)
*)


$Assumptions = Element[delta, Reals]
delta ∈ Reals

(* eqn A10, with M indicating matrix and V indicating vector:*)
pauseM =
  ( f * Exp[-kba * td] + (1 - f) * Exp[-r1aTd]   -f * Exp[-kba * td] + f * Exp[-r1aTd] );
  (     -Exp[-kba * td] + Exp[-r1aTd]                     Exp[-kba * td]              )
pauseV = ( 1 - Exp[-r1aTd] );
         ( 1 - Exp[-r1aTd] )

 pause[v_] := pauseM.v + pauseV;



(* za is just Exp[-R1p*pw](zai-za_ss)+z_ss.  zb is sum of R1p,
R1pfast, and damped cos and sin terms, each with za and zb
 coefficients (ending in 'C') and an offset (ending in 'O'). *)
(* eqns A12 and A13: *)
pulseM = (                                                        Exp[-R1pPw]
         ( R1pZaC * Exp[-R1pPw] + R1pfastZaC * Exp[-R1pfastPw] + cosZaC * Exp[-R2pbPw] * Co
pulseV = (                                                zaCWss * (1 - Exp[-R1pPw])
         ( zbCWss + R1pO * Exp[-R1pPw] + R1pfastO * Exp[-R1pfastPw] + cosO * Exp[-R2pbPw] *
pulse[v_] := pulseM.v + pulseV;
```

```
bothM = pulseM.pauseM;
bothV = pulseM.pauseV + pulseV;

(* eqn A15: *)
ss = Inverse[IdentityMatrix[2] - bothM].bothV;
MatrixForm[ss]
```

$$\left(\begin{array}{l} \frac{}{\left(1-e^{-R1pPw}\left(e^{-r1aTd}(1-f)+e^{-kba\,td}f\right)\right)\left(1-\left(e^{-r1aTd}f-e^{-kba\,td}f\right)\left(e^{-R1pfastPw}R1pfastZaC+e^{-R1pPw}R1pZaC+cosZaC\,e^{-R2pbPw}Cos[rabiBpw]+e^{-R2p}\right.\right.} \\ \frac{\left(1-e^{-R1pPw}\left(e^{-r1aTd}(1-f)+e^{-kba\,td}f\right)\right)\left(e^{-R1p}\right.}{\left(1-e^{-R1pPw}\left(e^{-r1aTd}(1-f)+e^{-kba\,td}f\right)\right)\left(1-\left(e^{-r1aTd}f-e^{-kba\,td}f\right)\left(e^{-R1pfastPw}R1pfastZaC+e^{-R1pPw}R1pZaC+cosZaC\,e^{-R2pbPw}Cos[rabiBpw]+e^{-R2p}\right.\right.} \end{array}\right.$$

```
(* WHAT FOLLOWS IS A KEY RESULT.  Calculate 1/ss (i.e. 1/Za) and take
 only the ZEROTH order in any combo of f, r1a*Td, and R1p*pw.  *)

(* m indicates minus, which is helpful since Mathematica
 likes to put expressions in the form Exp[ variable ] *)

ss0inv =
 Simplify[Normal[Series[1 / ss[[1, 1]] //. {kba td → -mKbaTd, R2pbPw → -mR2pbPw,
       R1pfastPw → -mR1pfastPw, f → ef e, r1aTd → er1aTd e, R1pPw → eR1pPw e},
     {e, 0, 0}]] //. {e → 1, ef → f, er1aTd → r1a * td, eR1pPw → R1p * pw}]
  (*1/ss expansion matches 1/(expansion of ss) *)
```

$$\Big(\left(f-e^{mKbaTd}f+pw\,R1p+r1a\,td\right)\left(1-e^{mKbaTd}\left(e^{mR1pfastPw}R1pfastZbC+R1pZbC+cosZbC\,e^{mR2pbPw}Cos[rabiBpw]+e^{mR2pbPw}sinZbC\,Sin[rabiBpw]\right)\right)+\left(f-e^{mKbaTd}f\right)\left(-e^{mR1pfastPw}R1pfastZaC-R1pZaC-cosZaC\,e^{mR2pbPw}Cos[rabiBpw]-e^{mR2pbPw}sinZaC\,Sin[rabiBpw]+\left(-1+e^{mKbaTd}\right)\left(e^{mR1pfastPw}R1pfastZbC+R1pZbC+cosZbC\,e^{mR2pbPw}Cos[rabiBpw]+e^{mR2pbPw}sinZbC\,Sin[rabiBpw]\right)\right)\Big)\Big/$$
$$\Big(-\left(-1+e^{mKbaTd}\right)f\left(e^{mR1pfastPw}R1pfastO+R1pO+zbCWss+cosO\,e^{mR2pbPw}Cos[rabiBpw]+e^{mR2pbPw}sinO\,Sin[rabiBpw]\right)+\left(r1a\,td+pw\,R1p\,zaCWss\right)\left(1-e^{mKbaTd}\left(e^{mR1pfastPw}R1pfastZbC+R1pZbC+cosZbC\,e^{mR2pbPw}Cos[rabiBpw]+e^{mR2pbPw}sinZbC\,Sin[rabiBpw]\right)\right)\Big)$$

```
(* Now use the same perturbative approach while including approximations
 for the amplitudes.  Use Torrey rates in amp approximations.  *)

(* For the following transforms,
start with eqns 23-27 (as derived in equations24to27.nb),
put in a normalized form, and then divide into components that
 will be multiplied by Zai (and are labelled with the ending ZaC),
that will be multiplied by Zbi (and are labelled with the ending ZbC),
and will not be multiplied by either
```

```
  Zai or Zbi (and are labelled with the ending O): *)

ampTransforms = {
   dDo → (alpha (1 + deltaA^2) (beta^2 + delta^2 + 2 beta k + k^2)) /
      ((1 + deltaA^2) (beta + k + (alpha + k) (delta^2 + (beta + k)^2))),
   dDzaC → (deltaA k (delta + delta^2 deltaA + deltaA (beta + k)^2) ) /
      ((1 + deltaA^2) (beta + k + (alpha + k) (delta^2 + (beta + k)^2))),
   dDzbC → 0,
   R1pfastO → -(1/gamma) alpha (a^2 + beta^2 + delta^2 + 2 beta k + k^2 - 2 a (beta + k)),
   R1pfastZaC → -((delta deltaA k + delta^2 deltaA^2 k + deltaA^2 k (-a + beta + k)^2) /
         ((1 + deltaA^2) gamma)),
   R1pfastZbC → -((-a delta^2 - a delta^2 deltaA^2 - a (-a + beta + k)^2 -
            a deltaA^2 (-a + beta + k)^2) / ((1 + deltaA^2) gamma)),
   cosO → -R1pfastO - dDo,
   cosZaC → -R1pfastZaC - dDzaC,
   cosZbC → 1 - R1pfastZbC - dDzbC,
   sinO → 1 / s * ( alpha + a * R1pfastO + b * cosO),
   sinZaC → 1 / s * ( (k * deltaA ^ 2 / (1 + deltaA ^ 2)) + a * R1pfastZaC + b * cosZaC ),
   sinZbC → 1 / s * ( - (alpha + k) + a * R1pfastZbC + b * cosZbC ),
   R1pO → - (R1pfastO + cosO) - zbCWss,
   R1pZaC → - (R1pfastZaC + cosZaC),
   R1pZbC → 1 - (R1pfastZbC + cosZbC)
  };
rateTransformsTorrey =
  { (* with alpha replaced by alpha+k and beta replaced by beta+k *)
   gamma → a * ( (b - a) ^ 2 + s ^ 2 ),
   a → (beta + k + (alpha + k) * delta ^ 2) / (1 + delta ^ 2),
   b → beta + k - 1 / 2 * (beta - alpha) / (1 + delta ^ 2),
   s → Surd[1 + delta ^ 2, 2]
  };
w1Transforms = { (* change from unitless terms to typical Bloch eqn terms *)
   alpha → r1b / w1,
   beta → r2b / w1,
   k → kba / w1,
   delta → dwb / w1,
   deltaA → dwa / w1
  };
ssTransfromsNormalized = {

   zbCWss → (alpha deltaA (beta^2 + delta^2 + 2 beta k + k^2) +
        k (delta + delta^2 deltaA + deltaA (beta + k)^2) zaCWss) /
```

```
      (deltaA (beta^2 (alpha + k) + alpha (delta^2 + k^2) + k (1 + delta^2 + k^2) +
          beta (1 + 2 alpha k + 2 k^2)))  (* Eqn 17 *)
  };  (* Idea:  Use Torrey (+k) rate values in amps, include steady state,
and then expand in terms of small f, r1a*td, R1p*pw, alpha.
 Plan:  first calc zbCWss in terms of alpha, etc. *)
ssTransfromsNormalizedZa = {
   zaCWss → deltaA^2 * r1a / (R1p * (1 + deltaA^2))
  };


(* expand to zero order in many parameters,
including beta.  Small parameter = e. Since beta is bigger than the others,
it is not justified, but it seems to be the only way to get something simple. *)
ss0invWithAmps =
  Normal[Series[1 / ss[[1, 1]] //. Join[{kba td → -mKbaTd, R2pbPw → -mR2pbPw,
        R1pfastPw → -mR1pfastPw, f → ef e, r1aTd → er1aTd e,
        R1pPw → eR1pPw e , alpha → ealpha e , beta → ebeta e}, ampTransforms,
       rateTransformsTorrey, ssTransfromsNormalized], {e, 0, 0}]] //.
   Join[{e → 1, ef → f, er1aTd → r1a * td, eR1pPw → R1p * pw, ealpha → alpha,
     ebeta → beta}, ssTransfromsNormalizedZa (* to get rid of zaCWss *)];

projectionFactor = (deltaA^2 pw / (1 + deltaA^2) + td) / (pw + td);

ss0invWithAmpsCancelProj = Simplify[ss0invWithAmps * projectionFactor]
```

$$-\left(\left(\left(1+\text{deltaA}^2\right)\left(\frac{\text{deltaA}^2\,\text{pw}}{1+\text{deltaA}^2}+\text{td}\right)\right.\right.$$

$$\left(\left(\text{f}-e^{\text{mKbaTd}}\,\text{f}+\text{pw}\,\text{R1p}+\text{r1a}\,\text{td}\right)\left(1-\text{delta}^2\left(-1+e^{\text{mKbaTd}+\text{mR1pfastPw}}\right)-\right.\right.$$

$$\left.e^{\text{mKbaTd}+\text{mR2pbPw}}\,\text{Cos}[\text{rabiBpw}]\right)+\frac{1}{\left(1+\text{deltaA}^2\right)\left(1+\text{delta}^2+\text{k}^2\right)}$$

$$\left(\text{f}-e^{\text{mKbaTd}}\,\text{f}\right)\left(\text{delta}\,\text{deltaA}\,(1+\text{delta}\,\text{deltaA})\,e^{\text{mR1pfastPw}}\left(1+\text{delta}^2+\text{k}^2\right)-\right.$$

$$\left(1+\text{delta}^2\right)\text{deltaA}\left(\text{delta}+\text{delta}^2\,\text{deltaA}+\text{deltaA}\,\text{k}^2\right)-$$

$$(\text{delta}-\text{deltaA})\,\text{deltaA}\,e^{\text{mR2pbPw}}\,\text{k}^2\,\text{Cos}[\text{rabiBpw}]+\left(1+\text{deltaA}^2\right)$$

$$\left(-1+e^{\text{mKbaTd}}\right)\left(1+\text{delta}^2+\text{k}^2\right)\left(\text{delta}^2\,e^{\text{mR1pfastPw}}+e^{\text{mR2pbPw}}\,\text{Cos}[\text{rabiBpw}]\right)+$$

$$\left.\left.\left.(\text{delta}-\text{deltaA})\,\text{deltaA}\,e^{\text{mR2pbPw}}\,\text{k}\,\text{Sin}[\text{rabiBpw}]\,\sqrt[2]{1+\text{delta}^2}\right)\right)\right)\Big/$$

$$\left(\text{r1a}\,(\text{pw}+\text{td})\left(\text{td}+\text{deltaA}^2\,(\text{pw}+\text{td})\right)\left(-1+\text{delta}^2\left(-1+e^{\text{mKbaTd}+\text{mR1pfastPw}}\right)+\right.\right.$$

$$\left.\left.e^{\text{mKbaTd}+\text{mR2pbPw}}\,\text{Cos}[\text{rabiBpw}]\right)\right)$$

```
(* eqn 28: *)
effPulsedR1p = Collect[
   Numerator[ss0invWithAmpsCancelProj]    * 1 / ((pw + td) (td + deltaA^2 (pw + td))
        (-1 + delta^2 (-1 + e^(mKbaTd+mR1pfastPw)) + e^(mKbaTd+mR2pbPw) Cos[rabiBpw]))  //.
    {pw → dc * ptr, td → ptr - pw}, {dc * R1p, (1 - dc) * r1a, f}, Simplify];
effPulsedR1pMatlab = Simplify[ effPulsedR1p //. w1Transforms]
(* Get rid of unitless notation for comparing
 with matlab calculations and paper equations*)
effPulsedR1a = Simplify[
  Denominator[ss0invWithAmpsCancelProj]  * 1 / ((pw + td) (td + deltaA^2 (pw + td))
      (-1 + delta^2 (-1 + e^(mKbaTd+mR1pfastPw)) + e^(mKbaTd+mR2pbPw) Cos[rabiBpw])) ]
```

$$\begin{aligned}&r1a - dc\, r1a + dc\, R1p - \Big(\Big(\left(-1+e^{mKbaTd}\right) f\, w1^2 \Big(-dwa^2\, dwb^2 + dwa\, dwb^3 - dwb^4 - dwa\, dwb^3\, e^{mR1pfastPw} + dwb^4\, e^{mR1pfastPw} -\\&\quad dwb^2\, kba^2 - dwa\, dwb\, e^{mR1pfastPw}\, kba^2 + dwb^2\, e^{mR1pfastPw}\, kba^2 - dwa^2\, w1^2 + dwa\, dwb\, w1^2 -\\&\quad 2\, dwb^2\, w1^2 - dwa\, dwb\, e^{mR1pfastPw}\, w1^2 + dwb^2\, e^{mR1pfastPw}\, w1^2 - kba^2\, w1^2 - w1^4 +\\&\quad e^{mR2pbPw}\left(dwa\, dwb\, kba^2 + dwa^2\left(dwb^2 + w1^2\right) + w1^2\left(dwb^2 + kba^2 + w1^2\right)\right) Cos[rabiBpw] +\\&\quad dwa\,(dwa - dwb)\, e^{mR2pbPw}\, kba\, w1\, Sin[rabiBpw] \sqrt[2]{1 + \frac{dwb^2}{w1^2}}\Big)\Big) \Big/ \Big(ptr\left(dwa^2 + w1^2\right)\\&\quad \left(dwb^2 + kba^2 + w1^2\right)\left(dwb^2\left(-1 + e^{mKbaTd+mR1pfastPw}\right) - w1^2 + e^{mKbaTd+mR2pbPw}\, w1^2\, Cos[rabiBpw]\right)\Big)\end{aligned}$$

$$r1a$$

```
(*eqn 28 comes first 3 terms from
 effPulsedR1pMatlab above plus the 4th term below: *)
effPulsedR1pMatlab4thTermNum =
 Collect[ 1 / w1^2 * Numerator[effPulsedR1pMatlab[[4]]],
  {Sin[rabiBpw], Cos[rabiBpw]}, Simplify]
effPulsedR1pMatlab4thTermDen =
 Simplify[ 1 / w1^2 * Denominator[effPulsedR1pMatlab[[4]]] ]
```

$$\begin{aligned}&\left(-1+e^{mKbaTd}\right) f \Big(dwa^2\left(dwb^2+w1^2\right) - \left(dwb^2\left(-1+e^{mR1pfastPw}\right) - w1^2\right)\left(dwb^2 + kba^2 + w1^2\right) +\\&\quad dwa\, dwb\left(dwb^2\left(-1+e^{mR1pfastPw}\right) - w1^2 + e^{mR1pfastPw}\left(kba^2 + w1^2\right)\right)\Big) - e^{mR2pbPw}\left(-1+e^{mKbaTd}\right)\\&\quad f\left(dwa\, dwb\, kba^2 + dwa^2\left(dwb^2 + w1^2\right) + w1^2\left(dwb^2 + kba^2 + w1^2\right)\right) Cos[rabiBpw] -\\&\quad dwa\,(dwa - dwb)\, e^{mR2pbPw}\left(-1+e^{mKbaTd}\right) f\, kba\, w1\, Sin[rabiBpw] \sqrt[2]{1 + \frac{dwb^2}{w1^2}}\end{aligned}$$

$$\begin{aligned}&\frac{1}{w1^2} ptr\left(dwa^2 + w1^2\right)\left(dwb^2 + kba^2 + w1^2\right)\\&\quad\left(dwb^2\left(-1 + e^{mKbaTd+mR1pfastPw}\right) - w1^2 + e^{mKbaTd+mR2pbPw}\, w1^2\, Cos[rabiBpw]\right)\end{aligned}$$

```
(* eqn 3: *)
effPulsedR1pDeltaAinf = Normal[Series[effPulsedR1p, {deltaA, Infinity, 0}]];
effPulsedR1pDeltaAinfMatlab = Simplify[ effPulsedR1pDeltaAinf //. w1Transforms]
```

$$r1a - dc\, r1a + dc\, R1p - \left(\left(-1 + e^{mKbaTd}\right) f\, w1^2 \left(\left(dwb^2 + w1^2\right) \left(-1 + e^{mR2pbPw} \mathrm{Cos}[rabiBpw]\right) + e^{mR2pbPw} kba\, w1\, \mathrm{Sin}[rabiBpw] \sqrt[2]{1 + \frac{dwb^2}{w1^2}}\right)\right) \Big/ \left(ptr \left(dwb^2 + kba^2 + w1^2\right) \left(dwb^2 \left(-1 + e^{mKbaTd+mR1pfastPw}\right) - w1^2 + e^{mKbaTd+mR2pbPw} w1^2 \mathrm{Cos}[rabiBpw]\right)\right)$$

```
ss0invWithAmpsDelta0DeltaAinf =
  Simplify[Limit[ss0invWithAmps /. delta → 0 , deltaA → Infinity]];
```

```
(* eqn 4: *)
effPulsedR1pDelta0DeltaAinf =
  Collect[1 / ((1 + k^2) (pw + td) (-1 + e^(mKbaTd+mR2pbPw) Cos[rabiBpw])) *
     Numerator[ss0invWithAmpsDelta0DeltaAinf] //.
    {pw → dc * ptr, td → ptr - pw}, {dc * R1p, (1 - dc) * r1a, f}, Simplify];
effPulsedR1pDelta0DeltaAinfMatlab = Simplify[
  effPulsedR1pDelta0DeltaAinf //. w1Transforms]
```

$$r1a - dc\, r1a + dc\, R1p - \left(\left(-1 + e^{mKbaTd}\right) f\, w1 \left(-w1 + e^{mR2pbPw} w1\, \mathrm{Cos}[rabiBpw] + e^{mR2pbPw} kba\, \mathrm{Sin}[rabiBpw]\right)\right) \Big/ \left(ptr \left(kba^2 + w1^2\right) \left(-1 + e^{mKbaTd+mR2pbPw} \mathrm{Cos}[rabiBpw]\right)\right)$$

```
effPulsedR1aDelta0DeltaAinf = 1 / ((1 + k^2) (pw + td) (-1 + e^(mKbaTd+mR2pbPw) Cos[rabiBpw])) *
  Denominator[ss0invWithAmpsDelta0DeltaAinf]
```

r1a