```
MODULE PmetricsTwoCptKaModel;

(* Same as TwoCptKaModel except that it is parameterizaed in terms of ka instead of ka - lambda1 *)

    IMPORT
        PmetricsPKModels,
        Math,
        PmetricsExpokitInterface,
        StdLog;

    TYPE
        TwoCptKaModel* = POINTER TO RECORD (PmetricsPKModels.PKModel) END;

    PROCEDURE (m: TwoCptKaModel) InitModel*;
    BEGIN
        m.nParameter := 11;
        m.F1Index := 5;
        m.tlag1Index := 8;
        m.nCmt := 3
    END InitModel;

    PROCEDURE (m: TwoCptKaModel) Pred1*(dt: REAL; IN parameter: PmetricsPKModels.ModelParameters;
        IN init, rate: ARRAY OF REAL; OUT pred: ARRAY OF REAL);
    (* Two compartment model with 1st order absorption *)
    (* Calculate amount in each compartment after dt time units where init = initial conditions and
    rate = rates into each compartment *)
    VAR
        a, alpha: ARRAY 3 OF REAL;
        CL, Q, V2, V3, ka, k10, k12, k21, ksum: REAL;
        i: INTEGER;
    BEGIN
(*
        StdLog.Real(dt); StdLog.String("   "); StdLog.Real(init[0]); StdLog.String("   ");
        StdLog.Real(rate[0]); .Ln;
*)
        CL := parameter.parameters[0];
        Q := parameter.parameters[1];
        V2 := parameter.parameters[2];
        V3 := parameter.parameters[3];
        ka := parameter.parameters[4];
        ASSERT((CL > 0) & (Q > 0) & (V2 > 0) & (V3 > 0) & (ka > 0) , 20);
        k10 := CL/V2;
        k12 := Q/V2;
        k21 := Q/V3;
        ksum := k10 + k12 + k21;
        alpha[0] := (ksum + Math.Sqrt(ksum*ksum-4.0*k10*k21))/2.0;
        alpha[1] := (ksum - Math.Sqrt(ksum*ksum-4.0*k10*k21))/2.0;
        alpha[2] := ka;

        pred[0] := 0;
        pred[1] := 0;
        pred[2] := 0;

        IF (init[0] # 0.0) OR (rate[0] # 0.0) THEN
            pred[0] := init[0] * Math.Exp(-ka*dt);
            pred[0] := pred[0] + rate[0] * (1.0 - Math.Exp(-ka*dt)) / ka;
            a[0] := ka * (k21 - alpha[0])/((ka-alpha[0])*(alpha[1]-alpha[0])) ;
            a[1] := ka * (k21 - alpha[1])/((ka-alpha[1])*(alpha[0]-alpha[1])) ;
            a[2] := -(a[0] + a[1]);
            pred[1] := pred[1] +
                PmetricsPKModels.PolyExp(dt,init[0],0,0,0,FALSE,a,alpha,3) +
                PmetricsPKModels.PolyExp(dt,0,rate[0],dt,0,FALSE,a,alpha,3);
            a[0] := ka * k12/((ka-alpha[0])*(alpha[1]-alpha[0])) ;
            a[1] := ka * k12/((ka-alpha[1])*(alpha[0]-alpha[1])) ;
```

```
        a[2] := -(a[0] + a[1]);
        pred[2] := pred[2] +
            PmetricsPKModels.PolyExp(dt,init[0],0,0,0,FALSE,a,alpha,3) +
            PmetricsPKModels.PolyExp(dt,0,rate[0],dt,0,FALSE,a,alpha,3);
    END;

    IF (init[1] # 0.0) OR (rate[1] # 0.0) THEN
        a[0] := (k21 - alpha[0])/(alpha[1]-alpha[0]) ;
        a[1] := (k21 - alpha[1])/(alpha[0]-alpha[1]) ;
        pred[1] := pred[1] +
            PmetricsPKModels.PolyExp(dt,init[1],0,0,0,FALSE,a,alpha,2) +
            PmetricsPKModels.PolyExp(dt,0,rate[1],dt,0,FALSE,a,alpha,2);
        a[0] := k12/(alpha[1]-alpha[0]) ;
        a[1] := -a[0];
        pred[2] := pred[2] +
            PmetricsPKModels.PolyExp(dt,init[1],0,0,0,FALSE,a,alpha,2) +
            PmetricsPKModels.PolyExp(dt,0,rate[1],dt,0,FALSE,a,alpha,2);
    END;

    IF (init[2] # 0.0) OR (rate[2] # 0.0) THEN
        a[0] := k21/(alpha[1]-alpha[0]) ;
        a[1] := -a[0];
        pred[1] := pred[1] +
            PmetricsPKModels.PolyExp(dt,init[2],0,0,0,FALSE,a,alpha,2) +
            PmetricsPKModels.PolyExp(dt,0,rate[2],dt,0,FALSE,a,alpha,2);
        a[0] := (k10 + k12 - alpha[0])/(alpha[1]-alpha[0]) ;
        a[1] := (k10 + k12 - alpha[1])/(alpha[0]-alpha[1]) ;
        pred[2] := pred[2] +
            PmetricsPKModels.PolyExp(dt,init[2],0,0,0,FALSE,a,alpha,2) +
            PmetricsPKModels.PolyExp(dt,0,rate[2],dt,0,FALSE,a,alpha,2);
    END;

END Pred1;

PROCEDURE (m:TwoCptKaModel) PredSS*(IN parameter: PmetricsPKModels.ModelParameters; amt, rate, ii:
REAL;
    cmt: INTEGER; OUT pred: ARRAY OF REAL);
(* Two compartment model with 1st order absorption *)
(* Calculate amount in each compartment at the end of a steady-state dosing interval or during
a steady-state constant input (if ii = 0) *)
VAR
    a, alpha: ARRAY 3 OF REAL;
    CL, Q, V2, V3, ka, k10, k12, k21, ksum: REAL;
    i: INTEGER;
BEGIN
    CL := parameter.parameters[0];
    Q := parameter.parameters[1];
    V2 := parameter.parameters[2];
    V3 := parameter.parameters[3];
    ka := parameter.parameters[4];
    ASSERT((CL > 0) & (Q > 0) & (V2 > 0) & (V3 > 0) & (ka > 0) , 20);
    k10 := CL/V2;
    k12 := Q/V2;
    k21 := Q/V3;
    ksum := k10 + k12 + k21;
    alpha[0] := (ksum + Math.Sqrt(ksum*ksum-4.0*k10*k21))/2.0;
    alpha[1] := (ksum - Math.Sqrt(ksum*ksum-4.0*k10*k21))/2.0;
    alpha[2] := ka;

    pred[0] := 0;
    pred[1] := 0;
    pred[2] := 0;

    IF rate = 0 THEN    (* bolus dose *)
```

```
    IF (cmt = 1) THEN
        a[0] := 0.0;
        a[1] := 0.0;
        a[2] := 1.0;
        pred[0] := PmetricsPKModels.PolyExp(ii,amt,0,0,ii,TRUE,a,alpha,3);
        a[0] := ka * (k21 - alpha[0])/((ka-alpha[0])*(alpha[1]-alpha[0])) ;
        a[1] := ka * (k21 - alpha[1])/((ka-alpha[1])*(alpha[0]-alpha[1])) ;
        a[2] := -(a[0] + a[1]);
        pred[1] := PmetricsPKModels.PolyExp(ii,amt,0,0,ii,TRUE,a,alpha,3);
        a[0] := ka * k12/((ka-alpha[0])*(alpha[1]-alpha[0])) ;
        a[1] := ka * k12/((ka-alpha[1])*(alpha[0]-alpha[1])) ;
        a[2] := -(a[0] + a[1]);
        pred[2] := PmetricsPKModels.PolyExp(ii,amt,0,0,ii,TRUE,a,alpha,3);

    ELSIF (cmt = 2) THEN
        a[0] := (k21 - alpha[0])/(alpha[1]-alpha[0]) ;
        a[1] := (k21 - alpha[1])/(alpha[0]-alpha[1]) ;
        pred[1] := PmetricsPKModels.PolyExp(ii,amt,0,0,ii,TRUE,a,alpha,2);
        a[0] := k12/(alpha[1]-alpha[0]) ;
        a[1] := -a[0];
        pred[2] := PmetricsPKModels.PolyExp(ii,amt,0,0,ii,TRUE,a,alpha,2);

    ELSE    (* cmt = 3 *)
        a[0] := k21/(alpha[1]-alpha[0]) ;
        a[1] := -a[0];
        pred[1] := PmetricsPKModels.PolyExp(ii,amt,0,0,ii,TRUE,a,alpha,2);
        a[0] := (k10 + k12 - alpha[0])/(alpha[1]-alpha[0]) ;
        a[1] := (k10 + k12 - alpha[1])/(alpha[0]-alpha[1]) ;
        pred[2] := PmetricsPKModels.PolyExp(ii,amt,0,0,ii,TRUE,a,alpha,2);
    END;

ELSIF ii > 0 THEN    (* multiple truncated infusions *)

    IF (cmt = 1) THEN
        a[0] := 0.0;
        a[1] := 0.0;
        a[2] := 1.0;
        pred[0] := PmetricsPKModels.PolyExp(ii,0,rate,amt/rate,ii,TRUE,a,alpha,3);
        a[0] := ka * (k21 - alpha[0])/((ka-alpha[0])*(alpha[1]-alpha[0])) ;
        a[1] := ka * (k21 - alpha[1])/((ka-alpha[1])*(alpha[0]-alpha[1])) ;
        a[2] := -(a[0] + a[1]);
        pred[1] := PmetricsPKModels.PolyExp(ii,0,rate,amt/rate,ii,TRUE,a,alpha,3);
        a[0] := ka * k12/((ka-alpha[0])*(alpha[1]-alpha[0])) ;
        a[1] := ka * k12/((ka-alpha[1])*(alpha[0]-alpha[1])) ;
        a[2] := -(a[0] + a[1]);
        pred[2] := PmetricsPKModels.PolyExp(ii,0,rate,amt/rate,ii,TRUE,a,alpha,3);

    ELSIF (cmt = 2) THEN
        a[0] := (k21 - alpha[0])/(alpha[1]-alpha[0]) ;
        a[1] := (k21 - alpha[1])/(alpha[0]-alpha[1]) ;
        pred[1] := PmetricsPKModels.PolyExp(ii,0,rate,amt/rate,ii,TRUE,a,alpha,2);
        a[0] := k12/(alpha[1]-alpha[0]) ;
        a[1] := -a[0];
        pred[2] := PmetricsPKModels.PolyExp(ii,0,rate,amt/rate,ii,TRUE,a,alpha,2);

    ELSE    (* cmt = 3 *)
        a[0] := k21/(alpha[1]-alpha[0]) ;
        a[1] := -a[0];
        pred[1] := PmetricsPKModels.PolyExp(ii,0,rate,amt/rate,ii,TRUE,a,alpha,2);
        a[0] := (k10 + k12 - alpha[0])/(alpha[1]-alpha[0]) ;
        a[1] := (k10 + k12 - alpha[1])/(alpha[0]-alpha[1]) ;
        pred[2] := PmetricsPKModels.PolyExp(ii,0,rate,amt/rate,ii,TRUE,a,alpha,2);
    END;
```

```
    ELSE    (* constant infusion *)

      IF (cmt = 1) THEN
        a[0] := 0.0;
        a[1] := 0.0;
        a[2] := 1.0;
        pred[0] := PmetricsPKModels.PolyExp(0,0,rate,INF,0,TRUE,a,alpha,3);
        a[0] := ka * (k21 - alpha[0])/((ka-alpha[0])*(alpha[1]-alpha[0])) ;
        a[1] := ka * (k21 - alpha[1])/((ka-alpha[1])*(alpha[0]-alpha[1])) ;
        a[2] := -(a[0] + a[1]);
        pred[1] := PmetricsPKModels.PolyExp(0,0,rate,INF,0,TRUE,a,alpha,3);
        a[0] := ka * k12/((ka-alpha[0])*(alpha[1]-alpha[0])) ;
        a[1] := ka * k12/((ka-alpha[1])*(alpha[0]-alpha[1])) ;
        a[2] := -(a[0] + a[1]);
        pred[2] := PmetricsPKModels.PolyExp(0,0,rate,INF,0,TRUE,a,alpha,3);

      ELSIF (cmt = 2) THEN
        a[0] := (k21 - alpha[0])/(alpha[1]-alpha[0]) ;
        a[1] := (k21 - alpha[1])/(alpha[0]-alpha[1]) ;
        pred[1] := PmetricsPKModels.PolyExp(0,0,rate,INF,0,TRUE,a,alpha,2);
        a[0] := k12/(alpha[1]-alpha[0]) ;
        a[1] := -a[0];
        pred[2] := PmetricsPKModels.PolyExp(0,0,rate,INF,0,TRUE,a,alpha,2);

      ELSE    (* cmt = 3 *)
        a[0] := k21/(alpha[1]-alpha[0]) ;
        a[1] := -a[0];
        pred[1] := PmetricsPKModels.PolyExp(0,0,rate,INF,0,TRUE,a,alpha,2);
        a[0] := (k10 + k12 - alpha[0])/(alpha[1]-alpha[0]) ;
        a[1] := (k10 + k12 - alpha[1])/(alpha[0]-alpha[1]) ;
        pred[2] := PmetricsPKModels.PolyExp(0,0,rate,INF,0,TRUE,a,alpha,2);
      END;

    END;

  END PredSS;

END PmetricsTwoCptKaModel.
```