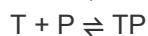


Scaffold protein titration motif

The model description

This particular motif describes one phosphorylation-desphosphorylation cycle (can be generalized to any futile cycles) with both kinase (K) and phosphatase (P) can be titrated by a scaffold protein (T).



The above reactions show a simple system composed of one scaffold protein, one kinase, one phosphatase and one substrate. Here we try to describe this simple system with differential equations following the mass action kinetics.

$$\frac{d[K]}{dt} = -k[1][K][S] + k[2][KS] + k[3][KS] - k[7][T][K] + k[8][TK],$$

$$\frac{d[P]}{dt} = -k[4][P][S_p] + k[5][PS_p] + k[6][PS_p] - k[9][T][P] + k[10][TP],$$

$$\frac{d[S]}{dt} = -k[1][K][S] + k[2][KS] + k[6][PS_p],$$

$$\frac{d[S_p]}{dt} = -k[4][P][S_p] + k[3][KS] + k[5][PS_p],$$

$$\frac{d[KS]}{dt} = k[1][K][S] - k[2][KS] - k[3][KS],$$

$$\frac{d[PS_p]}{dt} = k[4][P][S_p] - k[5][PS_p] - k[6][PS_p],$$

$$\frac{d[T]}{dt} = -k[7][T][K] + k[8][TK] - k[9][T][P] + k[10][TP] + k[11]k_d - k_d[T],$$

$$\frac{d[TK]}{dt} = k[7][T][K] - k[8][TK],$$

$$\frac{d[TP]}{dt} = k[9][T][P] - k[10][TP].$$

And the system needs to follow these conservation equations:

$$[K] + [KS] + [TK] = [K_{\text{tot}}],$$

$$[P] + [PS_p] + [TP] = [P_{\text{tot}}],$$

$$[S] + [S_p] + [KS] + [PS_p] = [S_{\text{tot}}],$$

$$[T] + [TK] + [TP] = [T_{\text{tot}}].$$

In the following section, we will solve the differential equations to understand the dynamics and behaviour of such system.

Understanding the dynamics of the simple system with input perturbations (numerical study)

Since, it is a bit difficult to solve the differential equations analytically. Here we try to study them numerically. By defining two different way to characterising the dynamics with scoring their temporal dynamics when presented with input signal perturbation (the changing of [T]). The quantification can be derived from the actually fitness functions for ultrasensitive response and adaptive response.

Then we save all the parameter sets as well as their score on ultrasensitivity and adaptation.

```
(NewKern) In[1]:= Clear["Global`*"];
SetDirectory[NotebookDirectory[]];
kd = 5;
des = {-k[1] * x[1][t] * x[3][t] + k[2] * x[5][t] + k[3] * x[5][t] -
k[7] * x[1][t] * x[7][t] + k[8] * x[8][t], -k[4] * x[2][t] * x[4][t] +
k[5] * x[6][t] + k[6] * x[6][t] - k[9] * x[2][t] * x[7][t] + k[10] * x[9][t],
-k[1] * x[1][t] * x[3][t] + k[2] * x[5][t] + k[6] * x[6][t],
-k[4] * x[2][t] * x[4][t] + k[3] * x[5][t] + k[5] * x[6][t],
k[1] * x[1][t] * x[3][t] - k[2] * x[5][t] - k[3] * x[5][t],
k[4] * x[2][t] * x[4][t] - k[5] * x[6][t] - k[6] * x[6][t],
-k[7] * x[1][t] * x[7][t] - k[9] * x[2][t] * x[7][t] +
k[8] * x[8][t] + k[10] * x[9][t] + k11[t] * kd - kd * x[7][t],
k[7] * x[1][t] * x[7][t] - k[8] * x[8][t],
k[9] * x[2][t] * x[7][t] - k[10] * x[9][t], 0};

init = {totK, totP, totS, 0, 0, 0, totT, 0, 0, 2.*10^-4}; (*init=
{tot[1],tot[2],tot[3],0.00001,0.00001,0.00001,totT,0.00001,0.00001};*)

(NewKern) In[6]:= AbsoluteTiming[
totK = 1; totP = 1; totS = 1;
stepNum = 5;
sampleSize = 100 000;

pars = {};
vars = Array[x, 9]; AppendTo[vars, k11];
dvars = Thread[Derivative[1][vars]];
SeedRandom[IntegerPart[SessionTime[]]];
ts = {};
For[num = 1, num <= sampleSize, num++,
Block[{k, T, ssthreshold}, k[n_] := k[n] = 10^(RandomReal[] * 6 - 3);
(*tot[n_]:=tot[n]=10^(RandomReal[]*4-3);*)
(*ksTest1=Array[k,10];*)
(*totT=1.*^3;*)
totT = 2.*10^(RandomReal[] * 4 - 3);

Block[{tPer, step},
step = 0;
tPer = {};
ssthreshold = 1.*^5;
(* Print[des]; *)
{sol} = NDSolve[{Through[dvars[t]] == des, Through[vars[0]] == init,
With[{df = Through[dvars[t]]}, WhenEvent[Norm[df] < ssthreshold,
AppendTo[tPer, t], step = step + 1, If[step > stepNum,
```

```

      "StopIntegration"], k11[t] → 10 * k11[t]]]],

vars, {t, 0, 200000}, MaxSteps → 10000];
ts = tPer;
If[Length[ts] == stepNum + 1 && AllTrue[ts, Positive],
x4 = Evaluate[x[4][ts - 0.001] /. sol];
xT =
Evaluate[(x[7][ts - 0.001] + x[8][ts - 0.001] + x[9][ts - 0.001]) /. sol];

us = Sqrt[((Abs[(x4[[4]] - x4[[3]])]) / totS) *
Min[((Abs[(x4[[4]] - x4[[3]])]) / Max[Abs[(x4[[3]] - x4[[1]])]],
0.001] + Abs[(x4[[4]] - x4[[3]])]) / Max[Abs[
(x4[[stepNum + 1]] - x4[[4]])], 0.001]) / 2) / 10.0, 1.0]];

ad = 0.0001;
For[i = 1, i ≤ stepNum, i++,
ad = ad *
Sqrt[(Min[(Max[Abs[Evaluate[x[4][Range[ts[[i]], ts[[i + 1]], 1]]] /.
sol] - Evaluate[x[4][ts[[i]]]] /. sol]] / (0.2 * totS)),
1.0] * Min[(0.01) / (Max[Abs[(x4[[i + 1]] - x4[[i]])] /
totS], 0.0001]), 1.0])];
];
ad = (ad / 0.0001)^(1 / (stepNum));

ks = Array[k, 10];
AppendTo[pars, Join[ks, {totT, totK, totP, totS, us, ad, num,
ks[[2]] + ks[[3]], ks[[5]] + ks[[6]], ks[[8]], ks[[10]]}]];
ks[[1]] ks[[4]] ks[[7]] ks[[9]]];
];
];
];
];
];
];

(*Plot@@{{{(x[7][t]+x[8][t]+x[9][t]),x[4][t]}/.sol},
Flatten@{t,x[1]["Domain"]/.sol},PlotLegends→{"Ttot","Sp"}}
ListPlot[Transpose@{xT,x4},PlotRange→{0,10}]*)
(*Print[pars];*)
transPars = Transpose[pars];
(*Export["saturationSampling.csv",transPars];*)
Export["unsaturationSampling.csv", transPars];

NDSolve:evcvmit: Eventlocation failed to converge to the requested accuracy or
precision within 100 iterations between t = 5171.71548546795 and t = 5224.48663768585>>
NDSolve:evcvmit: Eventlocation failed to converge to the requested accuracy or
precision within 100 iterations between t = 102.7677544401377 and t = 103.46704195804945>>
NDSolve:evcvmit: Eventlocation failed to converge to the requested accuracy or
precision within 100 iterations between t = 175.3462992344635 and t = 176.18102386941726>>

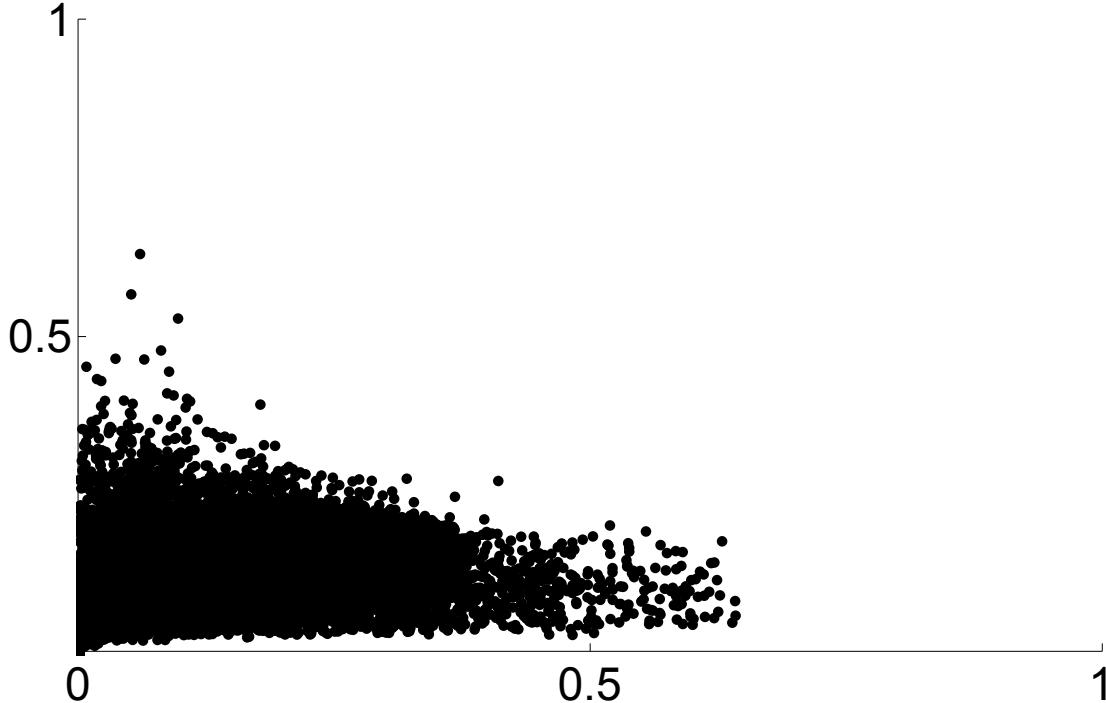
General::stop: Further output of NDSolve:evcvmit will be suppressed during this calculation>>

(NewKern) Out[6]= {26821.2, Null}

```

```
(NewKern) In[524]:= ListPlot[Transpose[{transPars[[15]], transPars[[16]]}],
 PlotRange -> {{0, 1}, {0, 1}},
 (*AxesLabel -> {"Ultrasensitive score", "Adaptive score"}, *)
 Ticks -> {{0, 0.5, 1}, {0.5, 1}}, PlotStyle -> {Thick, PointSize[0.01]},
 PlotTheme -> "Monochrome", PlotLabel -> None,
 LabelStyle -> {24, GrayLevel[0]}, ImageSize -> Large]
```

(NewKern) Out[524]=



```
(NewKern) In[10]:= maxAndIndex[a_] :=
 {#, First@SparseArray[UnitStep[a - #]] ["AdjacencyLists"]} & @Max@a
```

(NewKern) In[11]:= maxAndIndex[transPars[[15]]]

```
(NewKern) Out[11]=
 {0.641223, 98236}
```

(NewKern) In[12]:= maxAndIndex[transPars[[16]]]

```
(NewKern) Out[12]=
 {0.631595, 31386}
```

```
(NewKern) In[13]:= usIndex = maxAndIndex[transPars[[15]]] // Last;
 adIndex = maxAndIndex[transPars[[16]]] // Last;
 pars[[usIndex]]
```

```
(NewKern) Out[15]=
 {0.18691, 0.11489, 956.486, 0.00117357, 1.19166, 0.306976,
 122.133, 0.00565569, 0.706665, 0.25641, 3.62406, 1, 1, 1, 0.641223,
 0.0599328, 98236, 5117.98, 1276.99, 0.0000463075, 0.362845}
```

(NewKern) In[16]:= pars[[adIndex]]

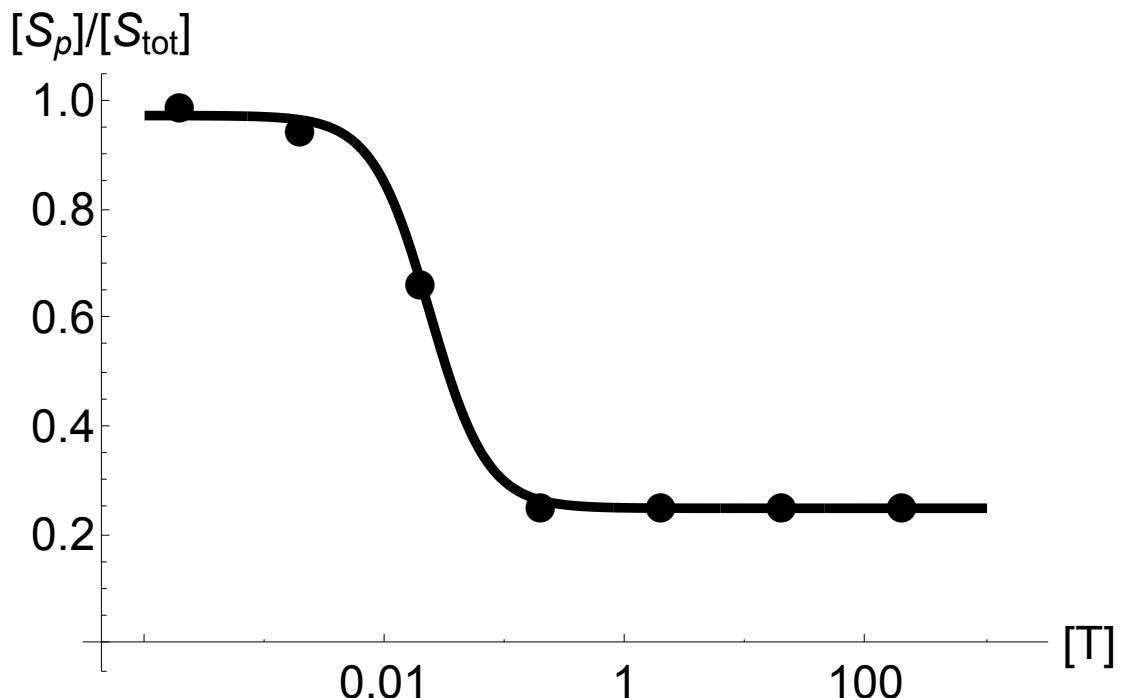
```
(NewKern) Out[16]=
 {328.589, 0.00116038, 240.675, 54.1912, 0.325441, 58.5094,
 0.201854, 0.00123987, 10.9987, 0.327257, 0.0116813, 1, 1, 1,
 0.0593904, 0.631595, 31386, 0.732454, 1.08569, 0.00614243, 0.0297541}
```

```
(NewKern) In[597]:= init = {totK, totP, totS, 0, 0, 0, totT, 0, 0, 2.*10^-4};
maxAndIndex[a_] :=
{#, First@SparseArray[UnitStep[a - #]]["AdjacencyLists"]}] &@Max@a
usIndex = maxAndIndex[transPars[[15]]] // Last;
adIndex = maxAndIndex[transPars[[16]]] // Last;
stepNum = 6;
maxPars = Solve[Array[k, 10] == pars[[usIndex]][[Range[10]]]];
totT = pars[[usIndex]][[11]];
Block[{tPer, step},
step = 0;
tPer = {};
ssthreshold = 1.*^-5;
(* Print[des]; *)
{sol} = NDSolve[{Through[dvars[t]] == des, Through[vars[0]] == init,
With[{df = Through[dvars[t]]},
WhenEvent[(Norm[df] < ssthreshold), {AppendTo[tPer, t], step = step + 1,
If[step > stepNum, "StopIntegration"], k11[t] \[Rule] 10*k11[t]}]]/.maxPars, vars, {t, 0, 200 000}, MaxSteps \[Rule] 10 000];
ts = tPer;
x4 = Evaluate[x[4][ts - 0.001] /. sol] / totS;
k11t = Evaluate[(k11[ts - 0.001]) /. sol];
];
fittedHill = FindFit[Transpose@{k11t, x4},
a + (b - a) * hillK / (hillK + x^(-n)), {a, b, hillK, n}, x]
Show[LogLinearPlot[a + (b - a) * hillK / (hillK + x^(-n)) /. fittedHill,
{x, 10^-4, 10^3}, (*Ticks \[Rule] {Automatic, {0, 0.5, 1}},*) PlotRange \[Rule] {-0.05, 1.05}, AxesLabel \[Rule] {"[T]", "[Sp]/[Stot]"}, PlotTheme \[Rule] "Monochrome", PlotStyle \[Rule] {Thickness[0.01]}],
ListLogLinearPlot[Transpose@{k11t, x4}, PlotTheme \[Rule] "Monochrome",
PlotMarkers \[Rule] {Automatic, 24}], PlotLabel \[Rule] None,
LabelStyle \[Rule] {24, GrayLevel[0]}, ImageSize \[Rule] Large]
```

(NewKern) Out[604]=

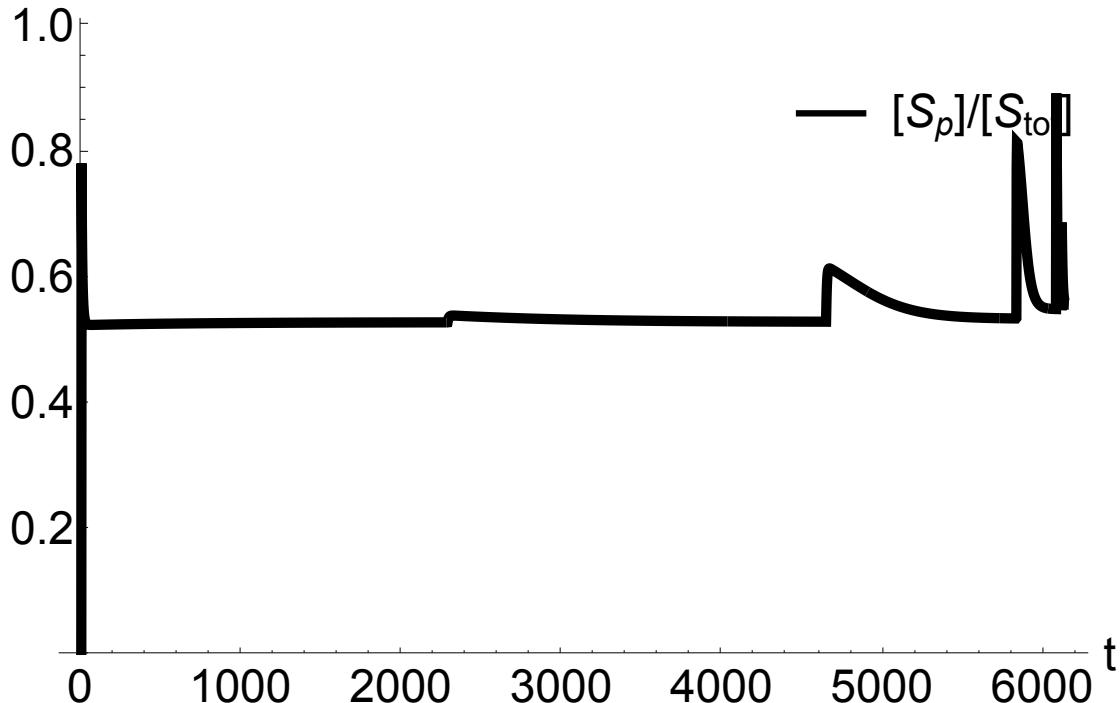
```
{a \[Rule] 0.974489, b \[Rule] 0.250516, hillK \[Rule] 945.506, n \[Rule] 1.82381}
```

(NewKern) Out[605]=



```
(NewKern) In[606]:= init = {totK, totP, totS, 0, 0, 0, totT, 0, 0, 2.*10^-4};
maxPars = Solve[Array[k, 10] == pars[[adIndex]][[Range[10]]]];
totT = pars[[adIndex]][[11]];
Block[{tPer, step},
  step = 0;
  tPer = {};
  ssthreshold = 1.*^5;
(* Print[des]; *)
{sol} = NDSolve[{Through[dvars[t]] == des, Through[vars[0]] == init,
  With[{df = Through[dvars[t]]},
    WhenEvent[Norm[df] < ssthreshold, {AppendTo[tPer, t], step = step + 1,
      If[step > stepNum, "StopIntegration"], k11[t] \[Rule] 10*k11[t]}]]] /.
  maxPars, vars, {t, 0, 200000}, MaxSteps \[Rule] 10000];
ts = tPer;
x4 = Evaluate[x[4][ts - 0.001] /. sol];
k11t = Evaluate[(k11[ts - 0.001]) /. sol];
];
Plot[{(x[4][t]/totS) /. sol}, {t, 0, ts[[stepNum]] - 0.01},
PlotLegends \[Rule] Placed[{"[Sp]/[Stot]", {0.85, 0.85}}, PlotRange \[Rule] {0, 1.01},
AxesLabel \[Rule] {"t"}, PlotTheme \[Rule] "Monochrome", PlotStyle \[Rule] {Thickness[0.01]},
PlotLabel \[Rule] None, LabelStyle \[Rule] {24, GrayLevel[0]}, ImageSize \[Rule] Large]
NDSolve::evcvmit: Event location failed to converge to the requested accuracy or
precision within 100 iterations. &gt; t = 4644.399374605779 &gt; dt = 4701.240731628203 >
```

(NewKern) Out[609]=

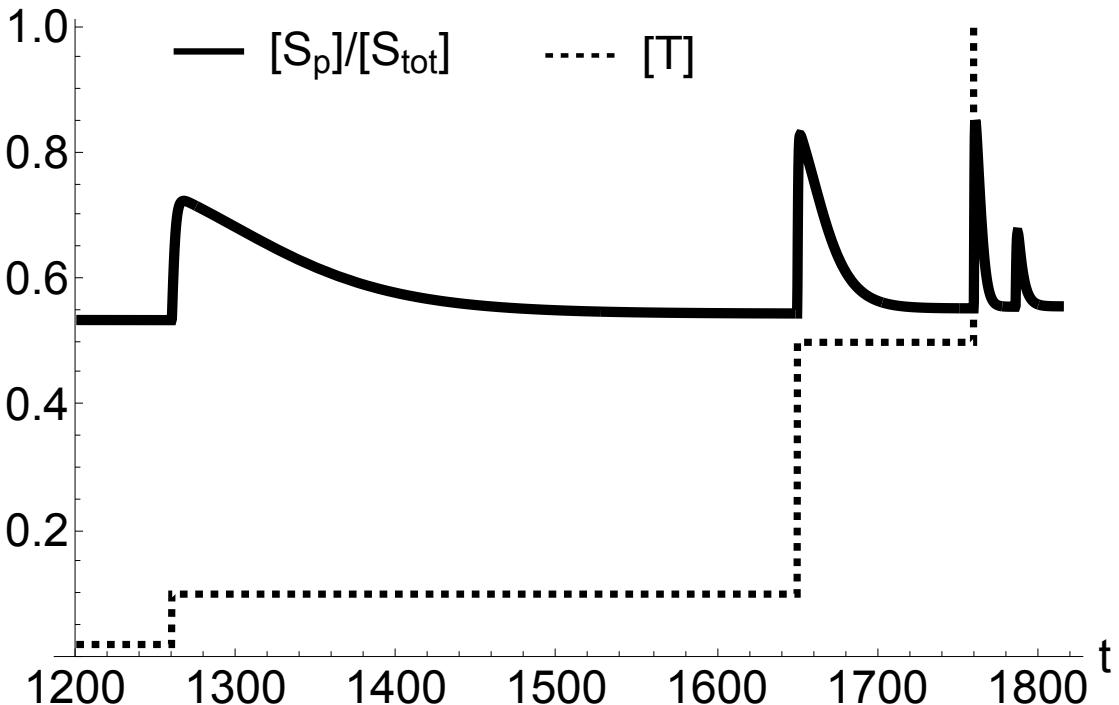


```
(NewKern) In[630]:= 
init = {totK, totP, totS, 0, 0, 0, totT, 0, 0, 0.02};
stepNum = 4;
maxPars = Solve[Array[k, 10] == pars[[adIndex]][[Range[10]]]];
totT = pars[[adIndex]][[11]];
Block[{tPer, step},
  step = 0;
  tPer = {};
  ssthreshold = 1.*^-5;
  (* Print[des]; *)
  {sol} = NDSolve[{Through[dvars[t]] == des, Through[vars[0]] == init,
    With[{df = Through[dvars[t]]},
      WhenEvent[Norm[df] < ssthreshold, {AppendTo[tPer, t], step = step + 1,
        If[step > stepNum, "StopIntegration"], k11[t] \[Rule] 5 * k11[t]}]]} /. maxPars, vars, {t, 0, 200 000}, MaxSteps \[Rule] 10 000];
  ts = tPer;
  x4 = Evaluate[x[4][ts - 0.001] /. sol];
  k11t = Evaluate[(k11[ts - 0.001]) /. sol];
];
Show[Plot[{{x[4][t] / totS} /. sol}, {t, 1200, ts[[stepNum + 1]] - 0.01},
  PlotLegends \[Rule] Placed[{"[Sp]/[Stot]", {0.25, 0.95}}, PlotRange \[Rule] {0, 1.01},
  AxesLabel \[Rule] {"t"}, PlotTheme \[Rule] "Monochrome", PlotStyle \[Rule] {Thickness[0.01]},
  PlotLabel \[Rule] None, LabelStyle \[Rule] {24, GrayLevel[0]}, ImageSize \[Rule] Large],
Plot[{{k11[t]} /. sol}, {t, 1200, ts[[stepNum + 1]] - 0.01},
  PlotLegends \[Rule] Placed[{"[T]", {0.55, 0.95}},
  PlotRange \[Rule] {0, 1.01}, AxesLabel \[Rule] {"t"}, PlotTheme \[Rule] "Monochrome",
  PlotStyle \[Rule] {Dashed, Thickness[0.007]}, PlotLabel \[Rule] None,
  LabelStyle \[Rule] {24, GrayLevel[0]}, ImageSize \[Rule] Large]]

```

(NewKern) Out[632]=
0.0116813

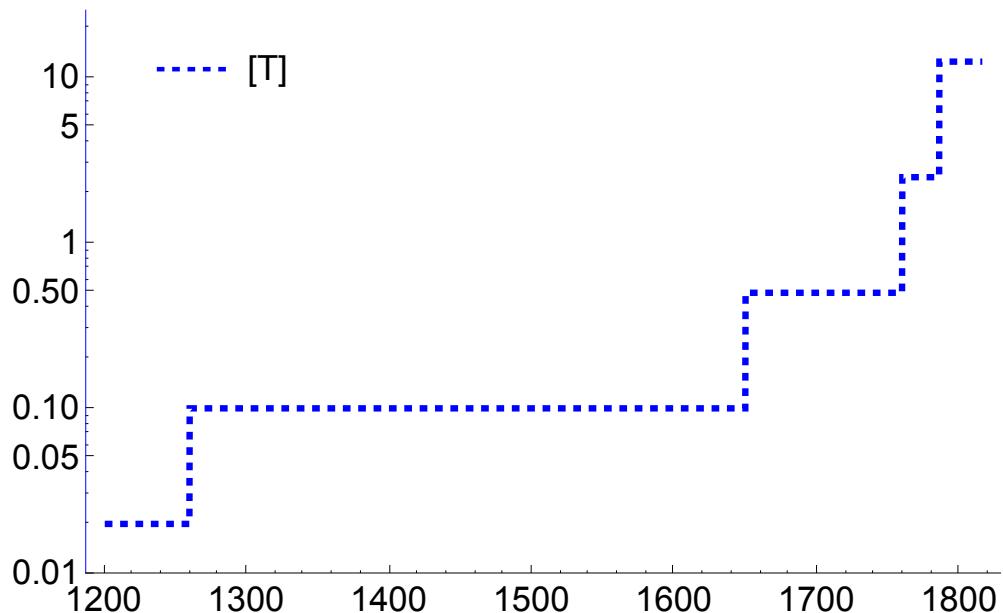
(NewKern) Out[634]=



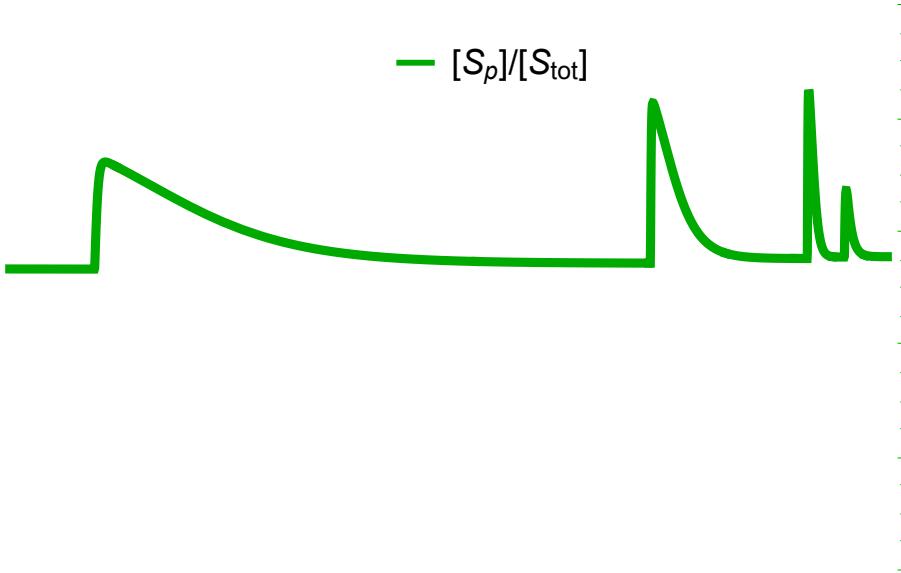
(NewKern) In[635]:=

```
input = LogPlot[{{k11[t]} /. sol}, {t, 1200, ts[[stepNum + 1]] - 0.01},  
PlotLegends → Placed[{"[T]"}, {0.15, 0.9}], PlotTheme → "Monochrome",  
PlotStyle → {Blue, Dashed, Thickness[0.007]}, Ticks → {},  
LabelStyle → {18, GrayLevel[0]}, ImageSize → Large,  
ImagePadding → 50, Frame → {True, True, False, False},  
FrameStyle → {Automatic, Blue, Automatic, Automatic}]
```

(NewKern) Out[635]=



```
(NewKern) In[637]:= output = Plot[{x[4][t] / totS} /. sol, {t, 1200, ts[[stepNum + 1]] - 0.01},  
PlotLegends → Placed[{"[Sp]/[Stot]", {0.55, 0.9}}, PlotRange → {0, 1},  
PlotStyle → {Darker[Green], Thickness[0.01]}, Ticks → {0, 0.5, 1},  
LabelStyle → {18, GrayLevel[0]}, ImageSize → Large, ImagePadding → 50,  
(*Axes→False,*)Frame → {False, False, False, True},  
FrameTicks → {None, None, None, {0, 0.5, 1}},  
FrameStyle → {Automatic, Automatic, Automatic, Darker[Green]}]  
(NewKern) Out[637]=
```



```
(NewKern) In[641]:=  
adPlot = Overlay[{input, output}];  
Export["scaffoldTitrationAlternativeUnsaturatedAd.eps", adPlot];  
Export["scaffoldTitrationAlternativeUnsaturatedAd.pdf", adPlot];  
(NewKern) Out[641]=
```

