



Mathematical Modelling and Scientific Computing in the Biosciences

| 15 May 2007

Lecture 6: Overview

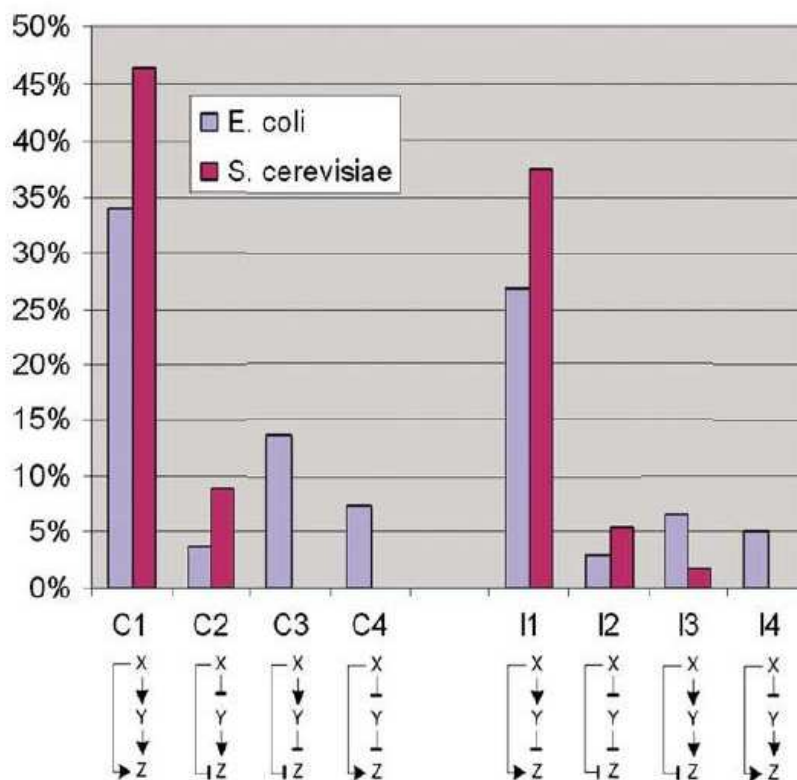
- **Dynamical Properties of Motifs in Biological Circuits**
– Incoherent Feed-Forward Loop (I1-FFL)
- **Temporal Ordering**



Motifs: Feed-Forward Loops

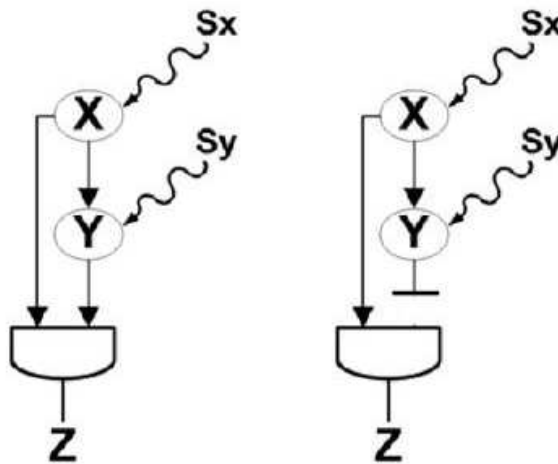
We looked at the most common type of feed-forward loop, the [coherent C1-FFL](#).

Now lets look at the 2nd most common type, the [incoherent I1-FFL](#).



Motifs: Feed-Forward Loops

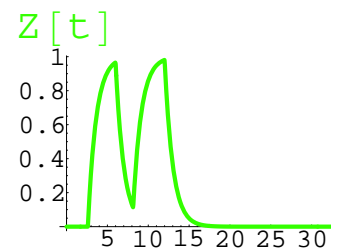
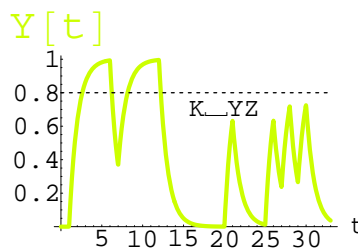
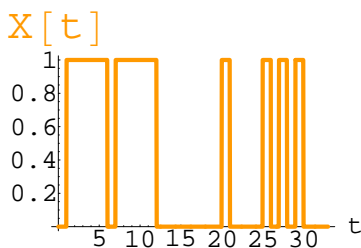
Coherent C1-FFL and incoherent I1-FFL



Motifs: Feed-Forward Loops

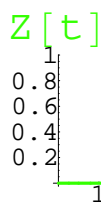
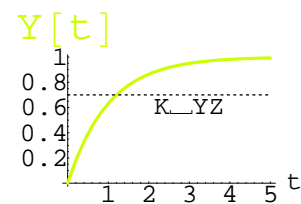
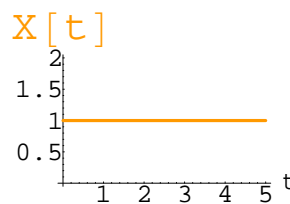
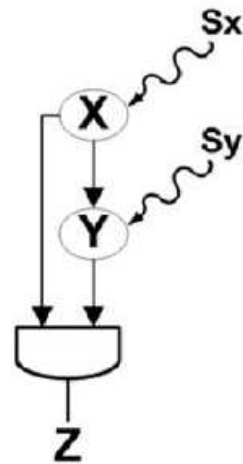
- **CIFFL solution:** filtering of brief **on** signals; well-resolved **off** signals.
- Arises from asymmetry to input on/off responses

```
In[71]:= XInput =
  {X[t] → UnitStep[t - 1] - UnitStep[t - 6] + UnitStep[t - 7] - UnitStep[t - 12] + UnitStep[t - 20] -
    UnitStep[t - 21] + UnitStep[t - 25] - UnitStep[t - 26] + UnitStep[t - 27] -
    UnitStep[t - 28] + UnitStep[t - 29] - UnitStep[t - 30]};
CIFFL_ODE_StepInput = {Derivative[1][Y][t] == β_Y UnitStep[X[t] - K_XY] - α_Y Y[t],
  Derivative[1][Z][t] == β_Z UnitStep[X[t] - K_XZ] UnitStep[Y[t] - K_YZ] - α_Z Z[t]} /. %;
tEnd = 33;
ConstantsNDSolve = {α_Y → 1, α_Z → 1, β_Y → 1, β_Z → 1, K_YZ → 0.8, K_XZ → 0.1, K_XY → 0.1};
CIFFLNumSoln = ({X[t] /. XInput, Y[t], Z[t]} /.
  NDSolve[CIFFL_ODE_StepInput /. %, Y[0] == 0, Z[0] == 0], {Y[t], Z[t]}, {t, 0, tEnd}) // Flatten;
PLabels = {"X[t]", "Y[t]", "Z[t]"};
SolnPlots = MapIndexed[Plot[#1, {t, 0, tEnd}, PlotStyle → {Hue[0.1 * #2[[1]]], Thickness[0.015]},
  AxesLabel → {"t", StyleForm[PLabels[#2[[1]]]}, FontColor → Hue[0.1 * #2[[1]]], FontSize → 16}],
  PlotRange → All, DisplayFunction → Identity] &, CIFFLNumSoln];
SolnPlotsNew = SolnPlots;
SolnPlotsNew[[2]] = Show[
  {SolnPlots[[2]], Graphics[{Dashing[{0.01, 0.02}], Line[{0, K_YZ}, {tEnd, K_YZ}] /. ConstantsNDSolve}],
  Graphics[Text["K_YZ", {20, -0.1 + K_YZ /. ConstantsNDSolve}]], DisplayFunction → Identity];
Show[GraphicsArray[{SolnPlotsNew}], DisplayFunction → $DisplayFunction, ImageSize → {500, 130}];
```

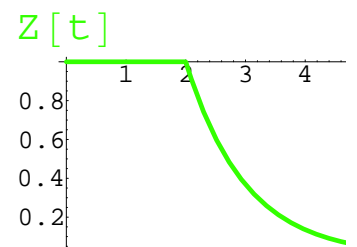
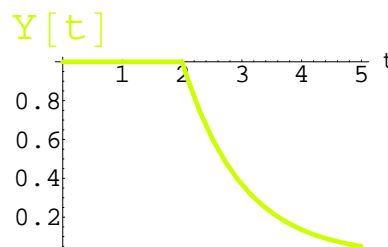
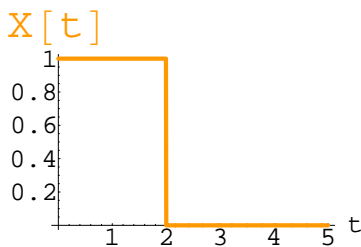


Motifs: Feed-Forward Loops

- Delay in $Z[t]$ when $X[t]: 0 \rightarrow 1$



- No delay when $X[t]: 1 \rightarrow 0$



Motifs: Feed-Forward Loops

- Due to the AND logic at Z , the equation for **CIFFL** takes the form:

```
In[81]:= C1FFL_ODE = {Derivative[1][Y][t] ==  $\beta_Y$  UnitStep[X[t] -  $K_{XY}$ ] -  $\alpha_Y$  Y[t],
  Derivative[1][Z][t] ==  $\beta_Z$  UnitStep[X[t] -  $K_{XZ}$ ] UnitStep[Y[t] -  $K_{YZ}$ ] -  $\alpha_Z$  Z[t]};
  % // ColumnForm // TraditionalForm
```

```
Out[82]//TraditionalForm=
```

$$Y'(t) = \beta_Y \theta(X(t) - K_{XY}) - \alpha_Y Y(t)$$

$$Z'(t) = \beta_Z \theta(X(t) - K_{XZ}) \theta(Y(t) - K_{YZ}) - \alpha_Z Z(t)$$

- In contrast, due to the AND NOT logic at output Z , **IIFFL** takes the form:

```
In[184]:=
I1FFL_ODE = {Derivative[1][Y][t] ==  $\beta_Y$  UnitStep[X[t] -  $K_{XY}$ ] -  $\alpha_Y$  Y[t],
  Derivative[1][Z][t] == ( $\beta_Z$  -  $\beta_{ZP}$ ) *
    (UnitStep[X[t] -  $K_{XZ}$ ] * (1 - UnitStep[Y[t] -  $K_{YZ}$ ])) +  $\beta_{ZP}$  -  $\alpha_Z$  Z[t]};
% // ColumnForm // TraditionalForm

Out[185]//TraditionalForm=

$$Y'(t) = \beta_Y \theta(X(t) - K_{XY}) - \alpha_Y Y(t)$$


$$Z'(t) = \beta_{ZP} + (\beta_Z - \beta_{ZP}) \theta(X(t) - K_{XZ}) (1 - \theta(Y(t) - K_{YZ})) - \alpha_Z Z(t)$$

```

where β_Z is the unrepressed transcription rate, $\beta_{ZP} \ll \beta_Z$ the repressed rate.

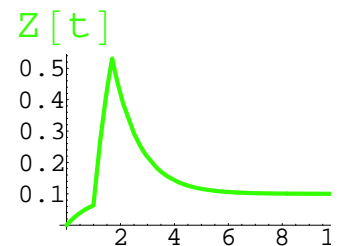
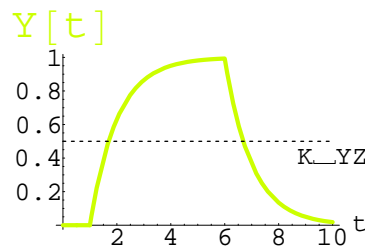
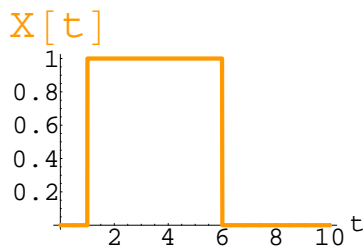


7 of 26

Motifs: Feed-Forward Loops

Let's try a simple step input for **I1FFL**:

```
In[186]:=
XInput = {X[t] -> UnitStep[t - 1] - UnitStep[t - 6]};
I1FFL_ODE_StepInput = I1FFL_ODE /. %;
tEnd = 10;
ConstantsNDSolve =
  { $\alpha_Y$  -> 1,  $\alpha_Z$  -> 1,  $\beta_Y$  -> 1,  $\beta_Z$  -> 1,  $\beta_{ZP}$  -> 0.1,  $K_{YZ}$  -> 0.5,  $K_{XZ}$  -> 0.5,  $K_{XY}$  -> 0.1};
I1FFLNumSoln = ({X[t] /. XInput, Y[t], Z[t]} /.
  NDSolve[{I1FFL_ODE_StepInput /. %, Y[0] == 0, Z[0] == 0}, {Y[t], Z[t]}, {t, 0, tEnd}]) // Flatten;
PLabels = {"X[t]", "Y[t]", "Z[t]"};
SolnPlots = MapIndexed[Plot[#1, {t, 0, tEnd}, PlotStyle -> {Hue[0.1 * #2[[1]]], Thickness[0.015]},
  AxesLabel -> {"t"}, StyleForm[PLabels[#2[[1]]], FontColor -> Hue[0.1 * #2[[1]]], FontSize -> 16}],
  PlotRange -> All, DisplayFunction -> Identity] &, I1FFLNumSoln];
SolnPlotsNew = SolnPlots;
SolnPlotsNew[[2]] = Show[
  {SolnPlots[[2]], Graphics[{Dashing[{0.01, 0.02}], Line[{0,  $K_{YZ}$ }, {tEnd,  $K_{YZ}$ }] /. ConstantsNDSolve]}],
  Graphics[Text[" $K_{YZ}$ ", {10, -0.1 +  $K_{YZ}$  /. ConstantsNDSolve}]]], DisplayFunction -> Identity];
Show[GraphicsArray[{SolnPlotsNew}], DisplayFunction -> $DisplayFunction, ImageSize -> {500, 130}];
```



8 of 26

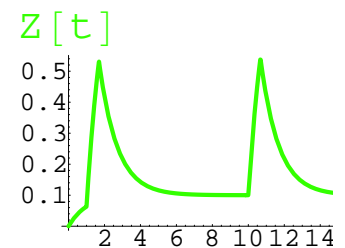
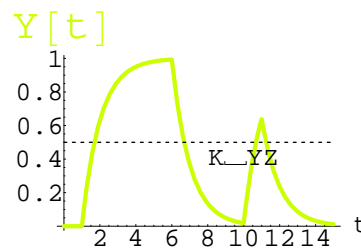
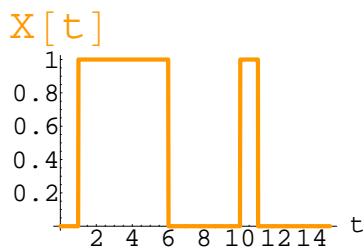
Motifs: Feed-Forward Loops

- The output $Z[t]$ essentially does not depend on the duration of the ON signal in $X[t]$:

```

In[196]:=
XInput = {X[t] -> UnitStep[t - 1] - UnitStep[t - 6] + UnitStep[t - 10] - UnitStep[t - 11]};
I1FFL_ODE_StepInput = I1FFL_ODE /. %;
tEnd = 15;
ConstantsNDSolve =
  { $\alpha_Y \rightarrow 1$ ,  $\alpha_Z \rightarrow 1$ ,  $\beta_Y \rightarrow 1$ ,  $\beta_Z \rightarrow 1$ ,  $\beta_{ZP} \rightarrow 0.1$ ,  $K_{YZ} \rightarrow 0.5$ ,  $K_{XZ} \rightarrow 0.5$ ,  $K_{XY} \rightarrow 0.1$ };
I1FFLNumSoln = ({X[t] /. XInput, Y[t], Z[t]} /.
  NDSolve[{I1FFL_ODE_StepInput /. %, Y[0] == 0, Z[0] == 0}, {Y[t], Z[t]}, {t, 0, tEnd}]) // Flatten;
PLabels = {"X[t]", "Y[t]", "Z[t]"};
SolnPlots = MapIndexed[Plot[#1, {t, 0, tEnd}], PlotStyle -> {Hue[0.1 * #2[[1]]], Thickness[0.015]},
  AxesLabel -> {"t", StyleForm[PLabels[#2[[1]]], FontColor -> Hue[0.1 * #2[[1]]], FontSize -> 16]},
  PlotRange -> All, DisplayFunction -> Identity] &, I1FFLNumSoln];
SolnPlotsNew = SolnPlots;
SolnPlotsNew[[2]] = Show[
  {SolnPlots[[2]], Graphics[{Dashing[{0.01, 0.02}], Line[{0, K_YZ}, {tEnd, K_YZ}] /. ConstantsNDSolve]}],
  Graphics[Text["K_YZ", {10, -0.1 + K_YZ /. ConstantsNDSolve}]], DisplayFunction -> Identity];
Show[GraphicsArray[{SolnPlotsNew}], DisplayFunction -> $DisplayFunction, ImageSize -> {500, 130}];

```

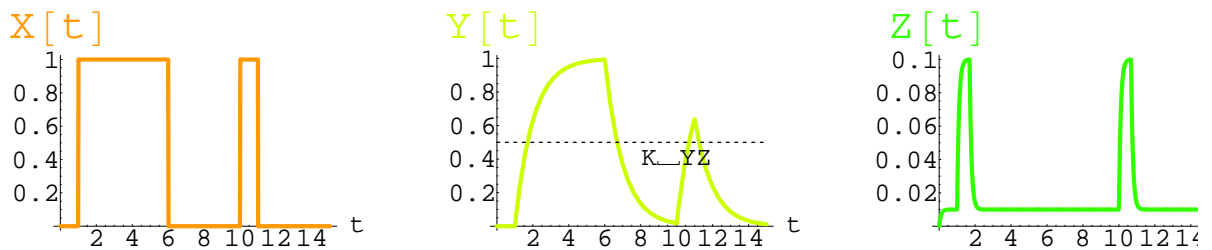


Motifs: Feed-Forward Loops

- Moreover, by increasing the degradation rate of gene Z (via α_Z), the peaks can become more **pulse-like**:

In[216]:=

```
XInput = {X[t] → UnitStep[t - 1] - UnitStep[t - 6] + UnitStep[t - 10] - UnitStep[t - 11]};
I1FFL_ODE_StepInput = I1FFL_ODE /. %;
tEnd = 15;
ConstantsNDSolve =
  {α_Y → 1, α_Z → 10, β_Y → 1, β_Z → 1, β_ZP → 0.1, K_YZ → 0.5, K_XZ → 0.5, K_XY → 0.1};
I1FFLNumSoln = ({X[t] /. XInput, Y[t], Z[t]} /.
  NDSolve[{I1FFL_ODE_StepInput /. %, Y[0] == 0, Z[0] == 0}, {Y[t], Z[t]}, {t, 0, tEnd}]) // Flatten;
PLabels = {"X[t]", "Y[t]", "Z[t]"};
SolnPlots = MapIndexed[Plot[#1, {t, 0, tEnd}], PlotStyle → {Hue[0.1 * #2[[1]]], Thickness[0.015]},
  AxesLabel → {"t", StyleForm[PLabels[#2[[1]]], FontColor → Hue[0.1 * #2[[1]]], FontSize → 16]},
  PlotRange → All, DisplayFunction → Identity] &, I1FFLNumSoln];
SolnPlotsNew = SolnPlots;
SolnPlotsNew[[2]] = Show[
  {SolnPlots[[2]], Graphics[{Dashing[{0.01, 0.02}], Line[{0, K_YZ}, {tEnd, K_YZ}] /. ConstantsNDSolve]}],
  Graphics[Text["K_YZ", {10, -0.1 + K_YZ /. ConstantsNDSolve}], DisplayFunction → Identity];
Show[GraphicsArray[{SolnPlotsNew}], DisplayFunction → $DisplayFunction, ImageSize → {500, 130}];
```



1 of 1

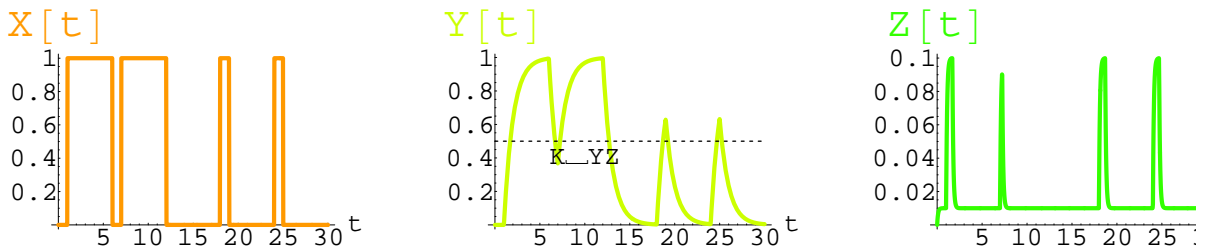
Motifs: Feed-Forward Loops

- For large α_Z , the system responds like a **pulse-generator**:

```

In[226]:=
XInput = {X[t] -> UnitStep[t - 1] - UnitStep[t - 6] + UnitStep[t - 7] - UnitStep[t - 12] +
  UnitStep[t - 18] - UnitStep[t - 19] + UnitStep[t - 24] - UnitStep[t - 25]};
I1FFL_ODE_StepInput = I1FFL_ODE /. %;
tEnd = 30;
ConstantsNDSolve =
  {α_Y -> 1, α_Z -> 10, β_Y -> 1, β_Z -> 1, β_ZP -> 0.1, K_YZ -> 0.5, K_XZ -> 0.5, K_XY -> 0.1};
I1FFLNumSoln = ({X[t] /. XInput, Y[t], Z[t]} /.
  NDSolve[{I1FFL_ODE_StepInput /. %, Y[0] == 0, Z[0] == 0}, {Y[t], Z[t]}, {t, 0, tEnd}]) // Flatten;
PLabels = {"X[t]", "Y[t]", "Z[t]"};
SolnPlots = MapIndexed[Plot[#1, {t, 0, tEnd}, PlotStyle -> {Hue[0.1 * #2[[1]]], Thickness[0.015]},
  AxesLabel -> {"t", StyleForm[PLabels[#2[[1]]], FontColor -> Hue[0.1 * #2[[1]]], FontSize -> 16}},
  PlotRange -> All, DisplayFunction -> Identity] &, I1FFLNumSoln];
SolnPlotsNew = SolnPlots;
SolnPlotsNew[[2]] = Show[
  {SolnPlots[[2]], Graphics[{Dashing[{0.01, 0.02}], Line[{0, K_YZ}, {tEnd, K_YZ}] /. ConstantsNDSolve}],
  Graphics[Text["K_YZ", {10, -0.1 + K_YZ /. ConstantsNDSolve}]]], DisplayFunction -> Identity];
Show[GraphicsArray[{SolnPlotsNew}], DisplayFunction -> $DisplayFunction, ImageSize -> {500, 130}];

```



11 of 26

Motifs: Feed-Forward Loops

- Consider gene Y undergoing the usual constant production and linear degradation:

```

In[325]:=
YEqn = I1FFL_ODE[[1]] /. {UnitStep[Input_] -> 1};
% /. t -> StyleForm[t, FontColor -> RGBColor[1, 0, 0]] // TraditionalForm

Out[326]//TraditionalForm=

$$Y'(t) = \beta_Y - \alpha_Y Y(t)$$


```

Setting the derivative to zero, find the possible steady-state solutions:

```

In[327]:=
YSteadyState =
  Solve[YEqn /. {Derivative[n_][x_][t_] -> 0}, Y[t]] /. {t -> Infinity} // Flatten // First

Out[327]=

$$Y[\infty] \rightarrow \frac{\beta_Y}{\alpha_Y}$$


```

- The solution for Y[t] can be easily solved: the solution involves exponential decay at rate α_Y :

```
In[321]:=
YEqnSoln =
  DSolve[{YEqn, Y[0] == 0}, Y[t], t] /. (YSteadyState /. Rule[a_, b_] -> Rule[b, a]) //
  Flatten // First // Simplify;
% /. t -> StyleForm[t, FontColor -> RGBColor[1, 0, 0]] // TraditionalForm

Out[322]//TraditionalForm=

$$Y(t) \rightarrow (1 - e^{-\alpha Y t}) Y(\infty)$$

```

12 of 26

Motifs: Feed-Forward Loops

```
In[328]:=
InitTimeAssumption = { Y[t] < K_YZ, X[t] > K_XZ };
InitZEqn = Simplify[IFFL_ODE[[2]], Assumptions -> %];
% // TraditionalForm
Null

Out[330]//TraditionalForm=

$$\beta_Z = \alpha_Z Z(t) + Z'(t)$$


In[331]:=
InitZSoln = DSolve[{InitZEqn, Z[0] == 0}, Z[t], t][[1, 1]] // FullSimplify;
% /. t -> StyleForm[t, FontColor -> RGBColor[1, 0, 0]] // TraditionalForm

Out[332]//TraditionalForm=

$$Z(t) \rightarrow \frac{\beta_Z - e^{-\alpha_Z t} \beta_Z}{\alpha_Z}$$

```

However, at t^* when $Y[t^*] = K_{YZ}$, gene Z is repressed: the ODE is then

```
In[375]:=
RepressedAssumption = { Y[t] > K_YZ, X[t] > K_XZ };
RepressedZEqn = Simplify[IFFL_ODE[[2]], Assumptions -> %];
% // TraditionalForm

Out[377]//TraditionalForm=

$$\beta_{ZP} = \alpha_Z Z(t) + Z'(t)$$

```

Let's now find out when (repression time for Z) t^* occurs:

```
In[378]:=
TStarEqn = Y[t] == K_YZ;

In[379]:=
Off[Solve::"ifun"];
t_StarSoln = (Solve[TStarEqn /. YEqnSoln, t] // Flatten // First // Simplify) /.
  {t -> t\[UnderBracket]Star}

Out[380]=

$$t_{\text{Star}} \rightarrow -\frac{\text{Log}\left[1 - \frac{K_{YZ}}{Y[\infty]}\right]}{\alpha_Y}$$

```

13 of 26

Motifs: Feed-Forward Loops

Now lets solve the equation for Z[t] after time t^* : the initial Z[t] solution at t^* is

```
In[381]:=
  InitZAt_t_Star = (InitZSoln // Last) /. t -> t_Star
```

```
Out[381]=
  
$$\frac{\beta_Z - e^{-t_{\text{Star}} \alpha_Z} \beta_Z}{\alpha_Z}$$

```

With the computed t^* substituted, this is:

```
In[382]:=
  InitZAt_t_Star = % /. t_StarSoln // FullSimplify
```

```
Out[382]=
  
$$-\frac{\beta_Z \left( -1 + \left( 1 - \frac{K_{YZ}}{Y[\infty]} \right)^{\frac{\alpha_Z}{\alpha_Y}} \right)}{\alpha_Z}$$

```

For simplicity, let us denote $Z[t^*]$ as Z_0

```
In[383]:=
  RepressedZSoln =
  FullSimplify[DSolve[{RepressedZEqn, Z[t_Star] == Z_0}, Z[t], t] // FullSimplify //
  Flatten // First;
  % /. t -> StyleForm[t, FontColor -> RGBColor[1, 0, 0]] // TraditionalForm
```

```
Out[384]//TraditionalForm=
  
$$Z(t) \rightarrow \frac{e^{\alpha_Z(t_{\text{Star}}-t)} (Z_0 \alpha_Z - \beta_Z P) + \beta_Z P}{\alpha_Z}$$

```

Which when using `t_StarSoln` explicitly, is:

```
In[385]:=
  FullRepressedZSoln = RepressedZSoln /. t_StarSoln // FullSimplify;
  % /. t -> StyleForm[t, FontColor -> RGBColor[1, 0, 0]] // TraditionalForm
```

```
Out[386]//TraditionalForm=
  
$$Z(t) \rightarrow \frac{e^{-\frac{\alpha_Z (\log[1 - \frac{K_{YZ}}{Y[\infty]}] + \alpha_Y t)}{\alpha_Y}} (Z_0 \alpha_Z - \beta_Z P) + \beta_Z P}{\alpha_Z}$$

```



Motifs: Feed-Forward Loops

Now let's look at the asymptotic limit of $t \rightarrow \infty$: try

```
In[387]:=
  ((RepressedZSoln // First) /. t -> Infinity) ->
  Limit[FullRepressedZSoln // Last, t -> Infinity]
```

```
Out[387]=
  
$$Z[\infty] \rightarrow \text{Limit} \left[ \frac{e^{-\frac{\alpha_Z (t \alpha_Y + \text{Log}[1 - \frac{K_{YZ}}{Y[\infty]}])}}{\alpha_Y}} (Z_0 \alpha_Z - \beta_Z P) + \beta_Z P}{\alpha_Z}, t \rightarrow \infty \right]$$

```

It didn't work because we need to know the signs of the constants. Extract the constants from the expression:

```
In[388]:=
  AllParam = Complement[Cases[RepressedZSoln // Last, _Symbol, Infinity] // Union, {t, e}];
  AllParamPosAssump = Map[# > 0 &, AllParam]

Out[389]=
  {t_Star > 0, Z0 > 0, alpha_Z > 0, beta_ZP > 0}
```

With the now with `Assumptions` → `AllParamPosAssump`, the limit calculation gives the result:

```
In[390]:=
  ZSteadyState = ((RepressedZSoln // First) /. t → Infinity) →
  Limit[FullRepressedZSoln // Last, t → Infinity, Assumptions → AllParamPosAssump]

Out[390]=
  Z[∞] →  $\frac{\beta_{ZP}}{\alpha_Z}$ 
```

15 of 26

Motifs: Feed-Forward Loops

Now let us manipulate `FullRepressedZSoln` to see how it goes from `Z0` to `Z[∞]`.

First, obtain replacement rule for `Z[∞]`

```
In[391]:=
  ZSteadyState /. Rule[a_, b_] → Rule[b, a]

Out[391]=
   $\frac{\beta_{ZP}}{\alpha_Z} \rightarrow Z[\infty]$ 
```

Algebraic manipulations give the result:

```
In[392]:=
  SimpRepressedZSoln = (Expand[#] & // @ RepressedZSoln) /. % // FullSimplify;
  % /. t → StyleForm[t, FontColor → RGBColor[1, 0, 0]] // TraditionalForm

Out[393]//TraditionalForm=
  Z(t) →  $e^{\alpha_Z(t_{Star}-t)} (Z0 - Z(\infty)) + Z(\infty)$ 
```

That is, it undergoes an exponential decay at rate α_Z after `t_Star`.

16 of 26

Motifs: Feed-Forward Loops

Now, let us find the response time for `Z` to increase to half its (final) steady-state value.

We equate $1/2 Z$

```
In[394]:=
ResponseTimeEqn = (InitZSoln // Last) == 1 / 2 * (ZSteadyState // Last);
% // TraditionalForm

Out[395]//TraditionalForm=

$$\frac{\beta_{-Z} - e^{-t\alpha_{-Z}} \beta_{-Z}}{\alpha_{-Z}} = \frac{\beta_{-ZP}}{2\alpha_{-Z}}$$


In[396]:=
ResponseTimeSoln = (Solve[ResponseTimeEqn, t] [[1, 1]] // FullSimplify) // ComplexExpand

Out[396]=

$$t \rightarrow -\frac{\text{Log}\left[1 - \frac{\beta_{-ZP}}{2\beta_{-Z}}\right]}{\alpha_{-Z}}$$

```

Let us now define the repression factor, $F = \beta_{-Z}/\beta_{-ZP}$ (which is $\gg 1$ given the assumptions)

```
In[403]:=
ResponseTime = Together //@ (ResponseTimeSoln /. {beta_ZP / beta_Z -> 1 / F})

Out[403]=

$$t \rightarrow -\frac{\text{Log}\left[\frac{-1+2F}{2F}\right]}{\alpha_{-Z}}$$

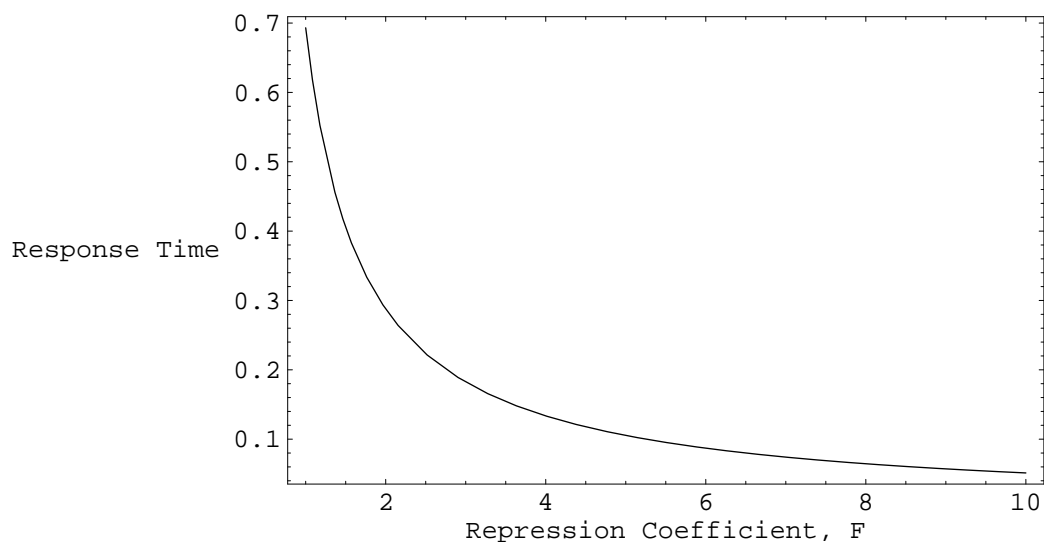
```

17 of 26

Motifs: Feed-Forward Loops

What does this response time look like?

```
In[417]:=
Plot[(ResponseTime // Last) /. alpha_Z -> 1, {F, 1, 10}, PlotRange -> All, Axes -> None,
Frame -> True, FrameLabel -> {"Repression Coefficient, F", "Response Time"},
RotateLabel -> False, ImageSize -> {400, 200}];
```



That is, the **higher** the $F = \beta_{-Z}/\beta_{-ZP}$, the **faster** the response.

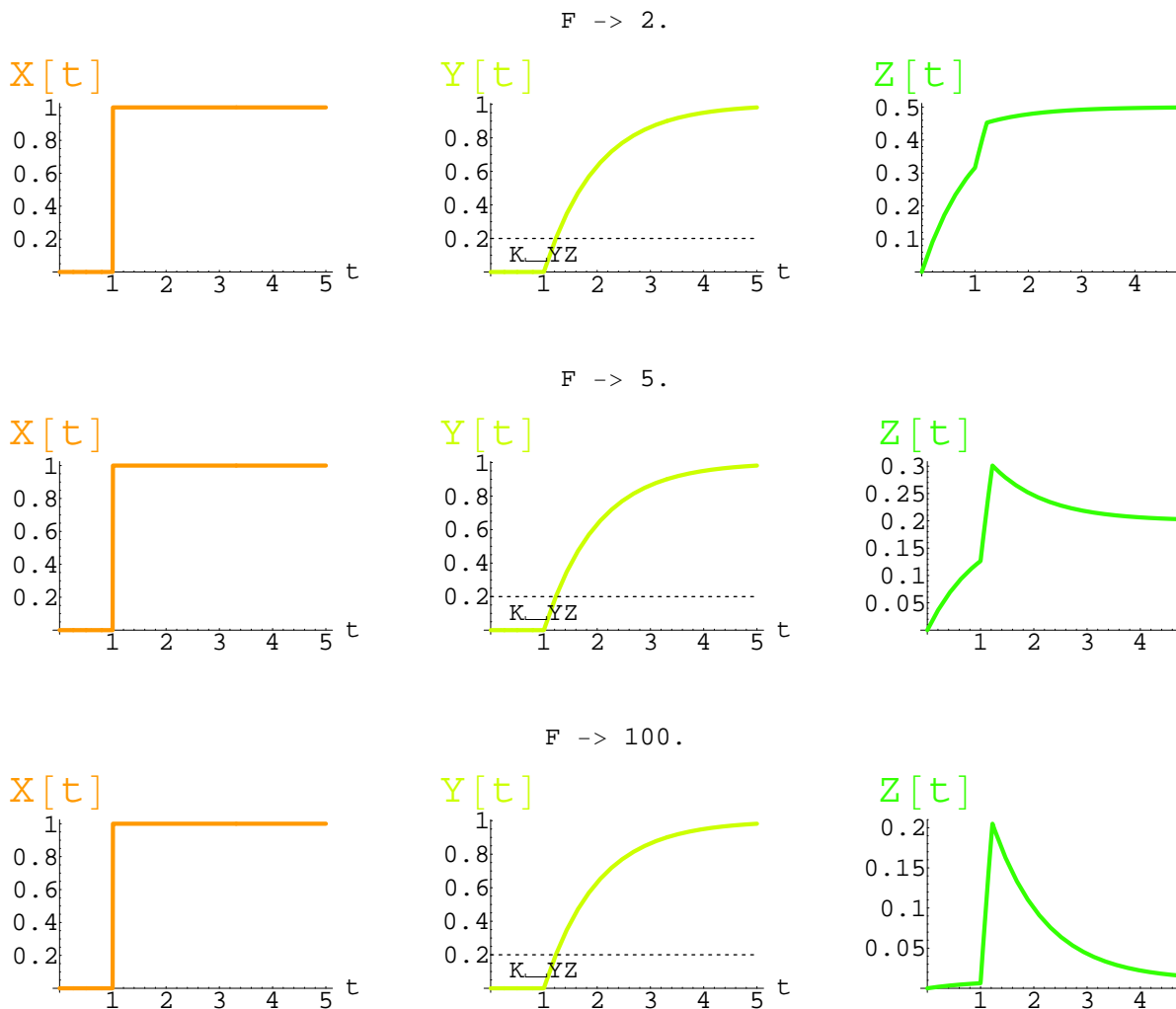
The speed-up of response time via repression, is reminiscent of **negative auto-regulation** (last lecture).

```
In[420]:=
ResponseTimeNoRepressionLimit = Limit[#, F -> 1] & //@ ResponseTime
```

```
Out[420]=
t ->  $\frac{\text{Log}[2]}{\alpha_Z}$ 
```

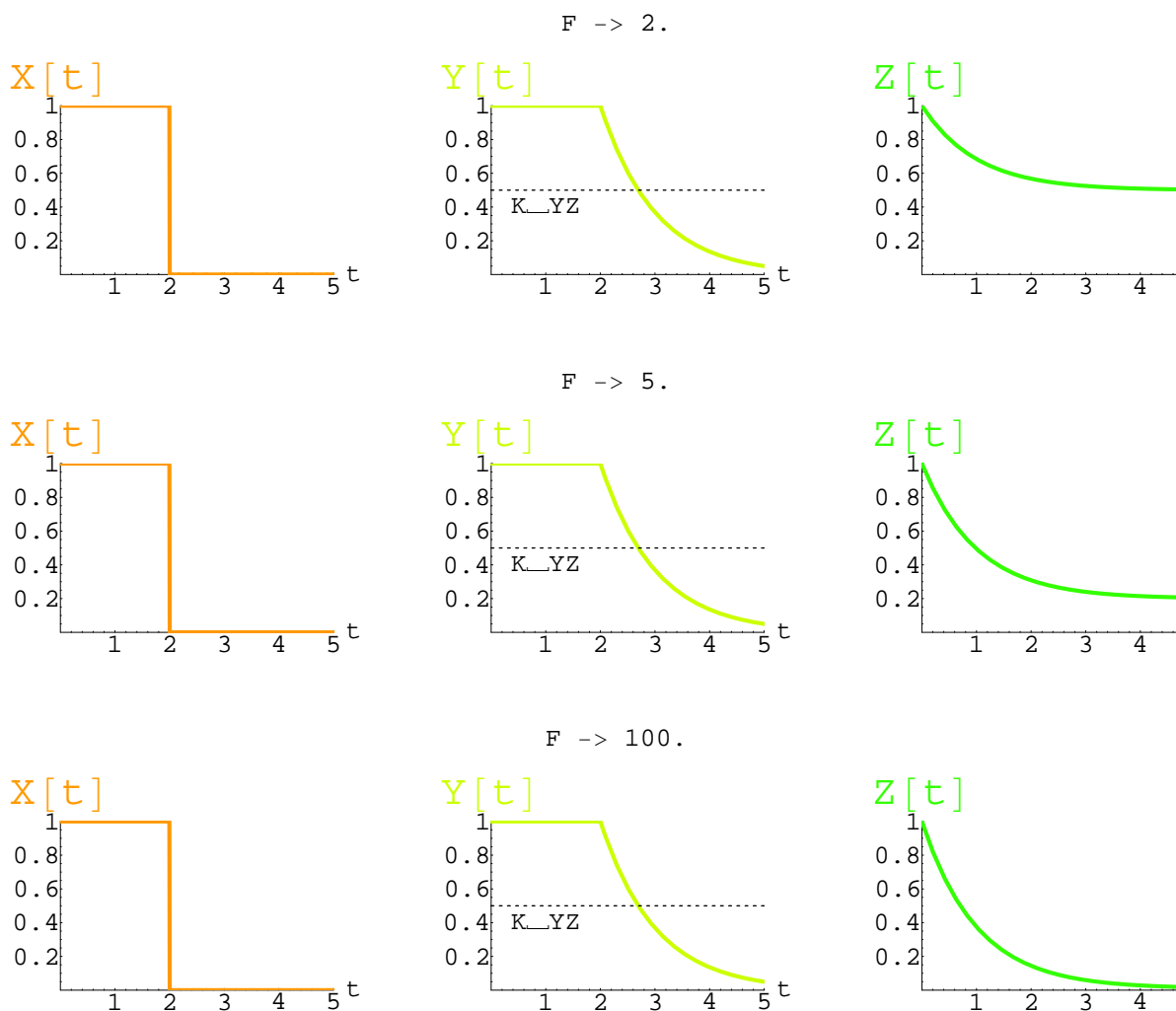
Motifs: Feed-Forward Loops

Speed-up for ON signal in X[t], for various repression coefficient F



Motifs: Feed-Forward Loops

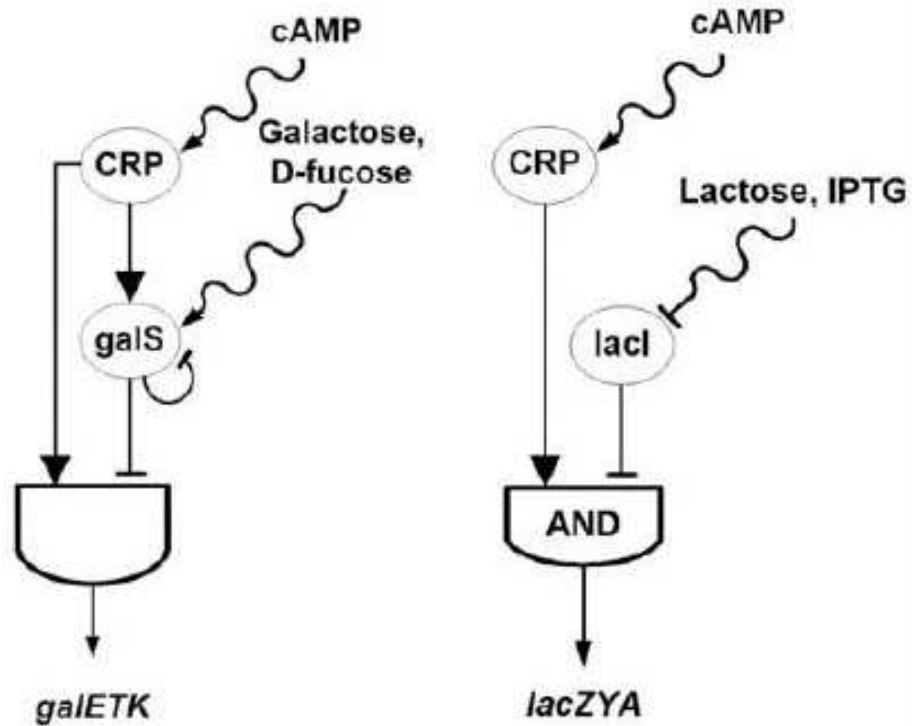
No speed-up for OFF signal in $X[t]$, for various repression coefficient F



Motifs: Feed-Forward Loops

Acceleration/pulse-like solution has been observed experimentally, for galatose utilization system:

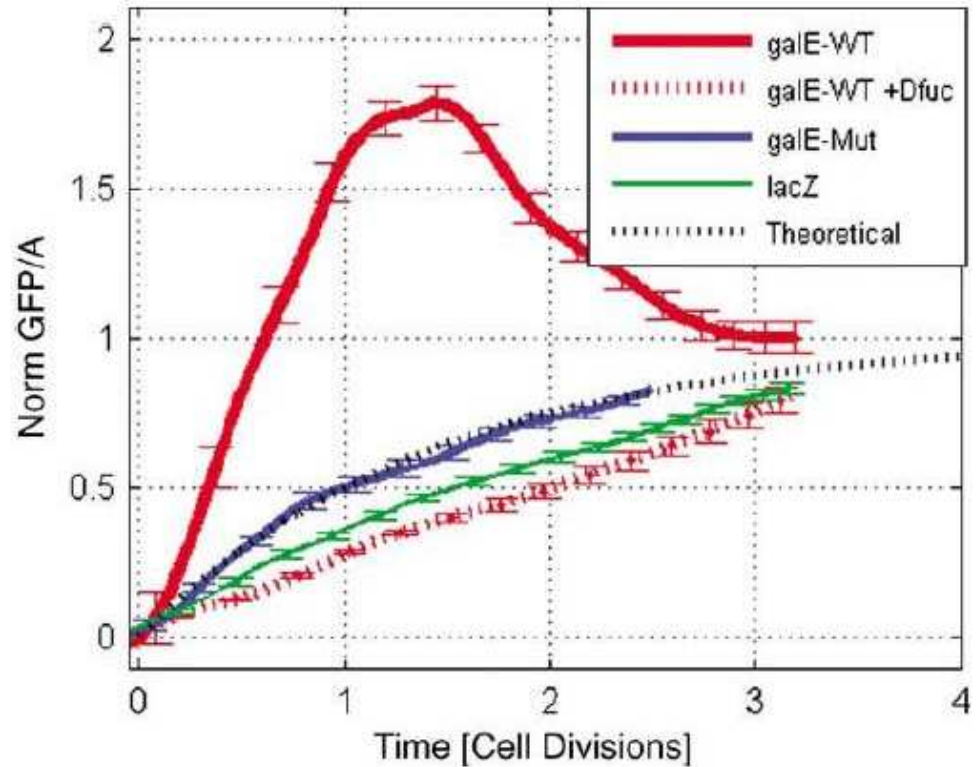
S. Mangan, S. Itzkovitz, A. Zaslaver and U. Alon, *J. Mol. Biol.* **235** (2006).



Motifs: Feed-Forward Loops

Without incoherent feed-forward loop, the lacZ system shows behavior approximated by $Z[t]/Z[\infty] = 1 - e^{-\alpha t}$.

In contrast, the behavior of wild-type galE shows (~3-fold) accelerated response and overshoot.



22 of 26

Conclusions:

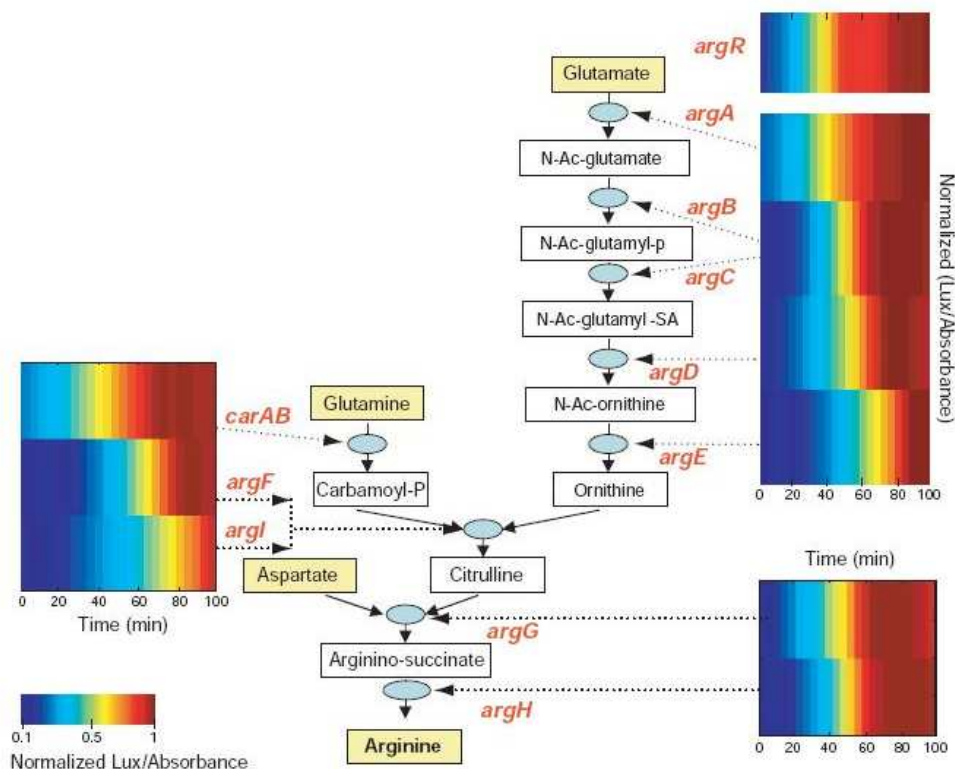
- **C1-FFL** can give act to filter brief (noisy) signals, and respond only to persistent ones.
- **I1-FFL** can
 - act as a pulse-generator
 - decrease response time to an ON signal (similar to negative auto-regulation)

23 of 26

Temporal Ordering

- Reaching production goal with minimal enzyme production: "just-in-time" transcription program : A. Zaslaver, A. E. Mayo et al, Nature Genetics **36** (2004).
- Activity profiles of of arginine biosynthesis operons:

```
In[1265]:=
Show[Import["TemporalOrder_ArginineBioSyn.jpg", "JPEG"], ImageSize -> {500, 300}];
```



- What motifs can give rise to these temporal orders?

Temporal Ordering

Consider regulator gene X controlling 3 genes: Z1, Z2, Z3 (e.g., genes responding to a single stress).

In particular, Z1, Z2, Z3 are activated, with activation thresholds K_{Z1} , K_{Z2} , K_{Z3} (with respect to X).

```
In[1499]:=
SIM_ODE = {X'[t] ==  $\beta_X$  UnitStep[t - 1] (1 - UnitStep[t - 5]) -  $\alpha_X$  X[t],
  Z1'[t] ==  $\beta_{Z1}$  * UnitStep[X[t] -  $K_{Z1}$ ] -  $\alpha_{Z1}$  * Z1[t],
  Z2'[t] ==  $\beta_{Z2}$  * UnitStep[X[t] -  $K_{Z2}$ ] -  $\alpha_{Z2}$  * Z2[t],
  Z3'[t] ==  $\beta_{Z3}$  * UnitStep[X[t] -  $K_{Z3}$ ] -  $\alpha_{Z3}$  * Z3[t]};
% // TableForm // TraditionalForm
```

```
Out[1500]//TraditionalForm=
 $X'(t) = \beta_X (1 - \theta(t - 5)) \theta(t - 1) - \alpha_X X(t)$ 
 $Z1'(t) = \beta_{Z1} \theta(X(t) - K_{Z1}) - \alpha_{Z1} Z1(t)$ 
 $Z2'(t) = \beta_{Z2} \theta(X(t) - K_{Z2}) - \alpha_{Z2} Z2(t)$ 
 $Z3'(t) = \beta_{Z3} \theta(X(t) - K_{Z3}) - \alpha_{Z3} Z3(t)$ 
```

Suppose $K_{Z1} < K_{Z2} < K_{Z3}$:

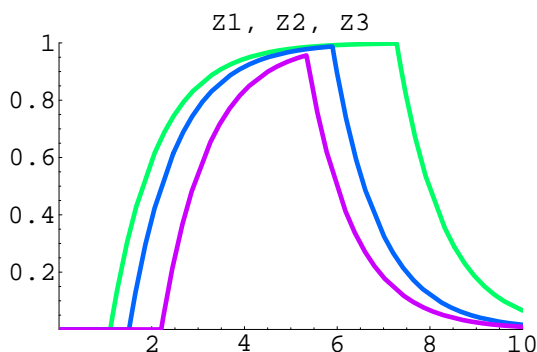
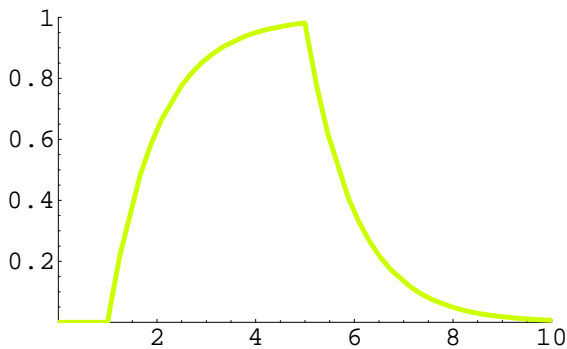
```
In[1501]:=
ConstantsRule =
  {K_Z1 → 0.1, K_Z2 → 0.4, K_Z3 → 0.7, K_X → 0.95, α_X → 1, α_Z1 → 1, α_Z2 → 1,
   α_Z3 → 1, β_X → 1, β_XP → 0.1, β_Z1 → 1, β_Z2 → 1, β_Z3 → 1, n → 8};
SIM_ODE /. ConstantsRule // TableForm // TraditionalForm
```

```
Out[1502]//TraditionalForm=
X'(t) = (1 - θ(t - 5))θ(t - 1) - X(t)
Z1'(t) = θ(X(t) - 0.1) - Z1(t)
Z2'(t) = θ(X(t) - 0.4) - Z2(t)
Z3'(t) = θ(X(t) - 0.7) - Z3(t)
```

25 of 26

Temporal Ordering

```
In[1509]:=
VarList = Cases[SIM_ODE, _Symbol[_Symbol], Infinity] // Union;
ICList = # == 0 & /@ VarList /. t → 0;
tEnd = 10;
SIM_Soln = NDSolve[{SIM_ODE /. ConstantsRule, ICList}, VarList, {t, 0, tEnd}] // Flatten;
SolnPlots =
  MapIndexed[Plot[#, {t, 0, tEnd}, PlotRange → {{0, tEnd}, {0, 1}}, PlotStyle → {Thickness[0.01],
    Hue[0.2 * First[#2]]}, DisplayFunction → Identity] &, VarList /. SIM_Soln];
Show[GraphicsArray[{{SolnPlots[[1]]}, {Show[{SolnPlots[[2]], SolnPlots[[3]], SolnPlots[[4]]},
  PlotLabel -> "Z1, Z2, Z3"]}]]];
```



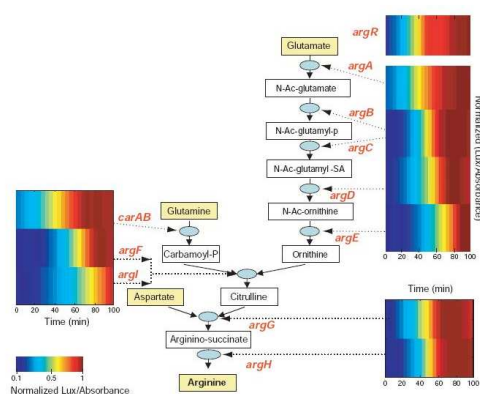
Switching-on behavior of Z's: Last-In-First-Out (LIFO)

26 of 26

Temporal Ordering

How to switch on genes Z1, Z2, Z3, controlled by X, that activations showing the First-In-First-Out (FIFO) order?

For instance, as seen in the activity profiles of of arginine biosynthesis operons:



Hint: generalize a feed-forward loop. We will consider this in an exercise.