

Mathematical Modelling and Scientific Computing in the Biosciences

| 8 May 2007

Lecture 5: Overview

- **Types of networks**
 - **Developmental transcription network**
 - **Signal transduction network**
 - **Sensory transduction**
 - **Neuronal network**
- **Dynamical Properties of Motifs in Biological Circuits**
 - **Mathematically controlled comparison**
 - **Negative auto-regulation vs. simple regulation**
 - **Feed-Forward Loop (FFL): information processing capability**

Types of Networks

There are several types of biological networks, each carrying out different function (or information processing) and hence have different dynamical properties. These is reflected in the **wiring properties** of the network; one way to look at how the various **network topologies** differ is to look at the **types** and **abundancies** of **network motifs**.

By only looking at the network topology, one abstracts only information such as X activates or inhibits Y, but not how it does at a biochemical level.

Sensory transduction

Sense and respond to the changing environment: e.g., levels of nutrients, stressful conditions. Dynamics typically on the scale of hours.

Signal transduction

Information processing of signaling proteins, functioning on the time scale of seconds to minutes.

Developmental transcription

In multi-cellular organisms, cells undergo **differentiation** process to change into the various cell types. Different specific sets of proteins are needed to form the various tissues, e.g, nerve, muscle. The process of (irreversible) differentiation is governed by developmental transcription networks.



3 of 30

Types of Networks

Sensory transduction

- Dynamical requirement: make **rapid** and **reversible decisions**.
- Example: Arabinose system of *E. coli*
 - Proteins that transport sugar arabinose into the cell to use it as a carbon source.
 - Decision based on the levels of the sugars: arabinose versus the superior energy source, glucose; logic: arabinose **AND NOT** glucose

Signal transduction

- Dynamical requirement: **fast dynamics**, form decisions from **large number of input stimuli**.
- Motifs: **multi-layer perceptrons** (perform computations on several input stimuli)
- Example: MAPK/ERK pathway
 - Coupling growth factors binding to cell surface receptors to intracellular responses.

Developmental transcription

- Dynamical requirements: **irreversible decisions**; **slower dynamics** (orchestrate developmental programs over several cell generations); **temporal ordering**.
- Motifs: display many of the motifs of sensory transduction networks. **Additional** motifs: **two-node positive feedback loops** (toggle-switch), **long cascades**
- Example: flagella assembly system in *E. coli*



4 of 30

Dynamical Properties of Motifs

Mathematically Controlled Comparison

- Motivation: why is there A and not B?
- Need to discover *characteristics* that distinguish between alternative designs
- How to make **mathematically controlled** comparison?
 - one design is chosen as the reference
 - for alternative designs:
 - maintain *internal equivalence* (e.g., values of biochemical parameters)
 - impose *external equivalence* (e.g., same final steady-states)
 - quantitatively compare *functional effectiveness* (e.g., how is the intermediate behavior of A better than B?)



5 of 30

Motifs: Simple Regulation VS. Negative Auto-Regulation

- Consider gene Y being up-regulated by gene X: $X \rightarrow Y$
- Assume:
 - X-activated production, $\begin{cases} \beta, & X > 0 \\ 0, & X = 0 \end{cases}$
 - constant degradation of protein Y, αY
- Differential equation:

```
In[492]:=
SimpleRegulationODE = Derivative[1][Y][t] ==  $\beta$  UnitStep[X] -  $\alpha$  Y[t];
% // TraditionalForm
```

```
Out[493]//TraditionalForm=
 $Y'(t) = \beta \theta(X) - \alpha Y(t)$ 
```

What is the asymptotic limit as $t \rightarrow \infty$?

```
In[494]:=
XInput = X  $\rightarrow$  1;
ODEIC = {SimpleRegulationODE /. XInput, Y[0] == 0};
SteadyStateSoln =
Solve[ODEIC /. Derivative[n_][Y_][t_]  $\rightarrow$  0, Y[t]] /. t  $\rightarrow$  Infinity // Flatten;
% // TraditionalForm
```

```
Out[497]//TraditionalForm=
```

$$\left\{ Y(\infty) \rightarrow \frac{\beta}{\alpha} \right\}$$

I.e., it is given by the ratio of production to degradation rates.

Motifs: Simple Regulation VS. Negative Auto-Regulation

```
In[509]:=
ODESoln = DSolve[ODEIC, Y[t], t] // Flatten; ODESoln // TraditionalForm
```

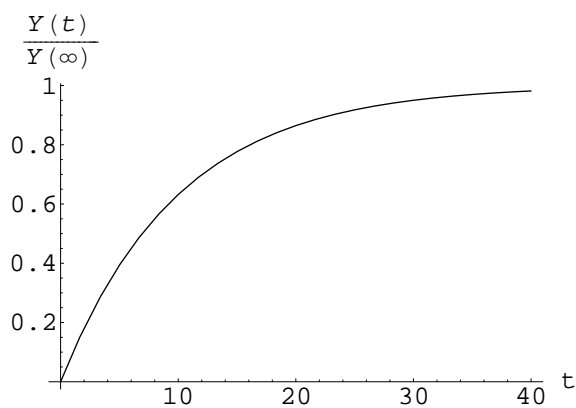
```
Out[509]//TraditionalForm=
{Y(t) ->  $\frac{e^{-t\alpha} (-1 + e^{t\alpha}) \beta}{\alpha}$ }
```

Rewrite this using steady-state solution $Y[\infty]$ as computed:

```
In[510]:=
YInf = SteadyStateSoln /. {Rule[LHS_, RHS_] -> Rule[RHS, LHS]};
ODESolnSimp = ODESoln /. % // Simplify; ODESolnSimp // TraditionalForm
```

```
Out[511]//TraditionalForm=
{Y(t) ->  $(1 - e^{-t\alpha}) Y(\infty)$ }
```

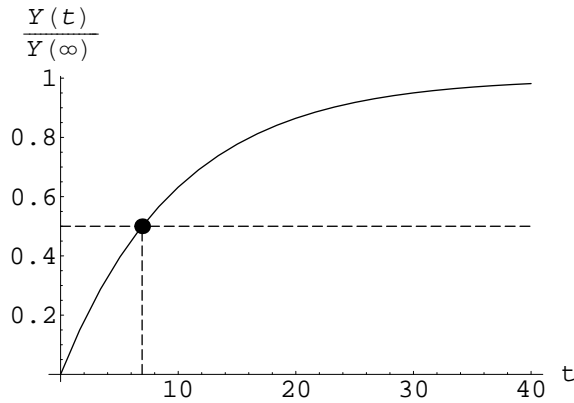
```
In[513]:=
PlotYRatio[ODESoln_, AlphaVal_] := Module[{PlotQuantity = Y[t] / Y[Infinity]},
  Plot[PlotQuantity /. ODESolnSimp /. {alpha -> AlphaVal},
    {t, 0, 40}, AxesLabel -> {"t", ToString[PlotQuantity // TraditionalForm]},
    ImageSize -> {300, 150}, DisplayFunction -> Identity];
Plot1 = Show[PlotYRatio[ODESoln, 0.1], DisplayFunction -> $DisplayFunction];
```



Motifs: Simple Regulation VS. Negative Auto-Regulation

Therefore, we have shown that $Y[t]$ increases towards $Y[\infty]$.

What is the [response time](#) (i.e., the time take to reach halfway between initial and final values)?



```
In[45]:= ResponseTimeEqn = (ODESolnSimp[[1]] // Last) == Y[∞] * 1 / 2
```

```
Out[45]= (1 - e-tα) Y[∞] ==  $\frac{Y[\infty]}{2}$ 
```

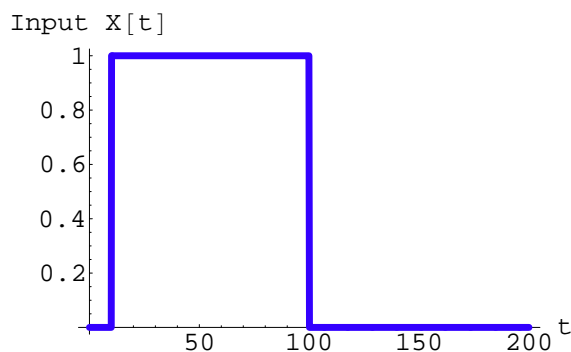
```
In[46]:= THalf = Reduce[ResponseTimeEqn, t, Reals] // Flatten
```

```
Out[46]= ( $\alpha \neq 0$  && t ==  $\frac{\text{Log}[2]}{\alpha}$ ) || Y[∞] == 0
```

That is, the response time with simple regulation depends only on the **degradation rate α** .

Motifs: Simple Regulation VS. Negative Auto-Regulation

```
In[233]:=
StepInput = UnitStep[t - 10] - UnitStep[t - 100];
StepInputPlot = Plot[%, {t, 0, 200}, PlotStyle -> {Hue[0.7], Thickness[0.015]},
  ImageSize -> {300, 130}, AxesLabel -> {"t", "Input X[t]"}];
```



```
In[172]:=
SimpleRegulationODE_StepInput = SimpleRegulationODE /. {UnitStep[X] -> StepInput}
```

```
Out[172]=
Y'[t] ==  $\beta (-\text{UnitStep}[-100 + t] + \text{UnitStep}[-10 + t]) - \alpha Y[t]$ 
```

In[173]:=

```
StepInputSoln = DSolve[%, Y[0] == 0], Y[t], t] /. YInf // Simplify // Flatten
```

Out[173]=

$$\{Y[t] \rightarrow \begin{cases} e^{-(10+t)\alpha} (-1 + e^{90\alpha}) Y[\infty] & t > 100 \\ (1 - e^{-(10+t)\alpha}) Y[\infty] & 10 < t \leq 100 \end{cases}\}$$



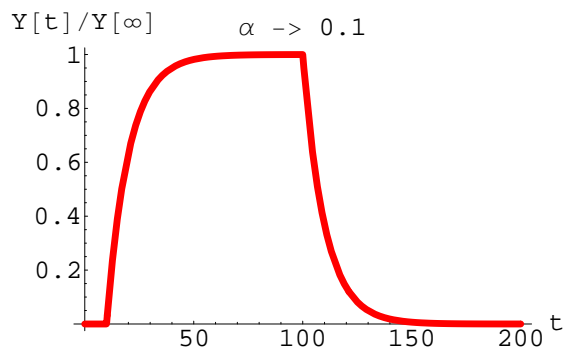
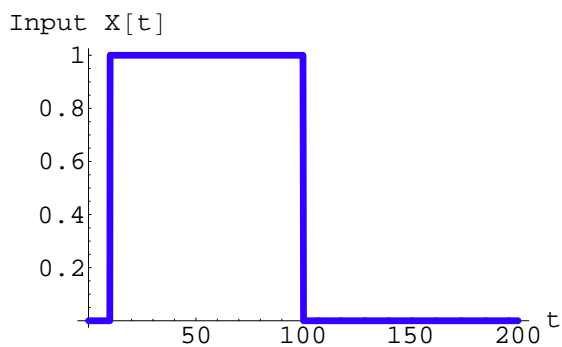
9 of 30

Motifs: Simple Regulation VS. Negative Auto-Regulation

In[202]:=

```
AlphaRule =  $\alpha \rightarrow 0.1$ ;
RespPlot[AlphaRule_] := Plot[Y[t]/Y[ $\infty$ ] /. StepInputSoln /. AlphaRule // Simplify,
{t, 0, 200}, PlotStyle -> {Hue[0.0], Thickness[0.015]},
ImageSize -> {300, 130}, PlotRange -> All, AxesLabel -> {"t", "Y[t]/Y[ $\infty$ ]"},
PlotLabel -> ToString[AlphaRule], DisplayFunction -> Identity];
```

```
Show[GraphicsArray[{{StepInputPlot}, {RespPlot[AlphaRule]}]];
```

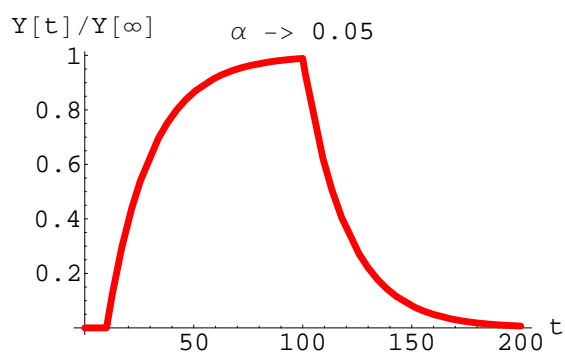
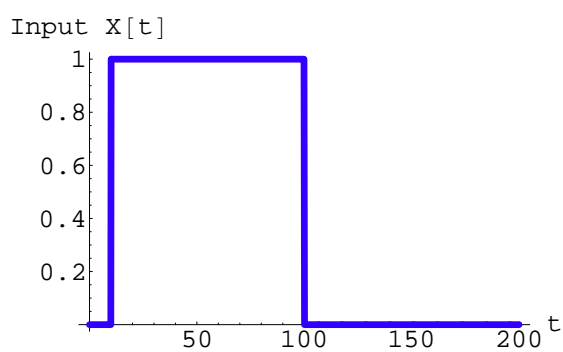


10 of 30

Motifs: Simple Regulation VS. Negative Auto-Regulation

```
In[213]:=
```

```
AlphaRule =  $\alpha \rightarrow 0.05$ ;  
Show[GraphicsArray[{{StepInputPlot}, {RespPlot[AlphaRule]}}]];
```



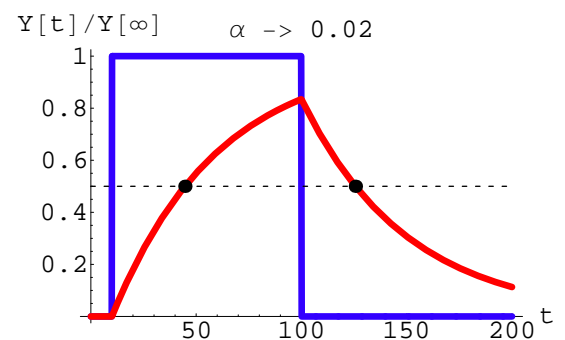
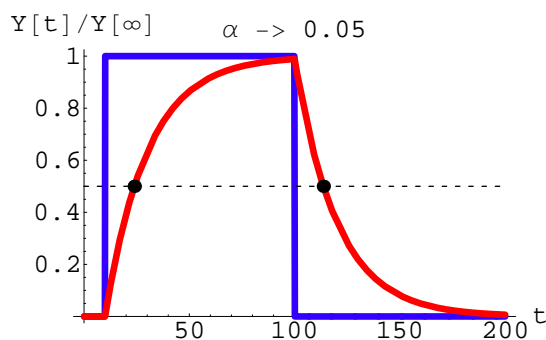
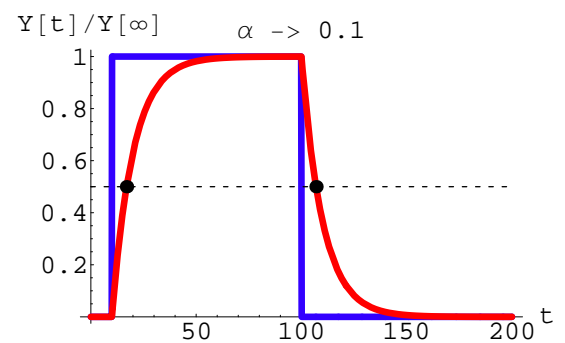
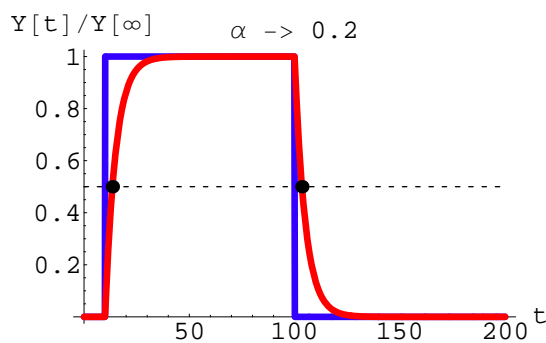
That is, we have shown the response time decreases as the degradation rate α increases.



Motifs: Simple Regulation VS. Negative Auto-Regulation

In[345]:=

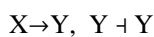
```
AlphaRuleList = { $\alpha \rightarrow 0.2$ ,  $\alpha \rightarrow 0.1$ ,  $\alpha \rightarrow 0.05$ ,  $\alpha \rightarrow 0.02$ };
Off[Reduce::"ratnz"];
Show[{{StepInputPlot, RespPlot[#],
Graphics[{{Dashing[{0.01, 0.02}], Line[{{0, 1/2}, {200, 1/2}}]}]},
Graphics /@ ({AbsolutePointSize[5.0], Point[{#, 1/2]}]} & /@ Cases[Reduce[
(Y[t]/Y[ $\infty$ ] /. StepInputSoln /. # // Simplify) = 1/2, Reals], _Real, Infinity]}]},
DisplayFunction -> Identity, PlotLabel -> ToString[#], AxesLabel -> {"t", "Y[t]/Y[ $\infty$ ]",
PlotRange -> All] & /@ AlphaRuleList;
Show[GraphicsArray[Partition[%, 2]], ImageSize -> {500, 280}];
```



12 of 30

Motifs: Simple Regulation VS. Negative Auto-Regulation

- Consider gene Y being regulated by gene X, as well as being **negatively auto-regulated** by itself:



- Assume:

-X-activated production, described by Hill-functions $\begin{cases} \beta/(1 + (Y/K)^n), & X > 0 \\ 0, & X = 0 \end{cases}$

-constant degradation of protein Y, αY

- Differential equation:

```
In[198]:=
NARregulationODE = Derivative[1][Y][t] ==  $\beta$  UnitStep[X] / (1 + (Y[t]/K)^n) -  $\alpha$  Y[t];
% // TraditionalForm
```

```
Out[199]//TraditionalForm=
```

$$Y'(t) = \frac{\beta \theta(X)}{\left(\frac{Y(t)}{K}\right)^n + 1} - \alpha Y(t)$$

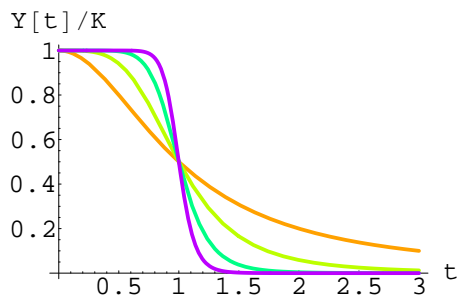
For ease of analysis, let's approximate the term $\left(\frac{Y(t)}{K}\right)^n + 1$, in the case n is large.



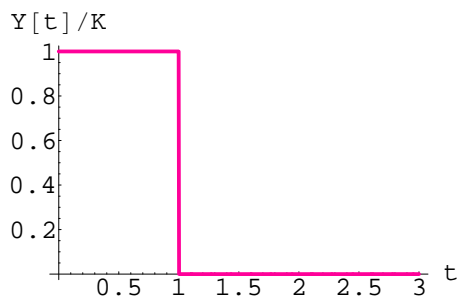
13 of 30

Motifs: Simple Regulation VS. Negative Auto-Regulation

```
In[200]:=
NARFunction[x_] := 1 / (x^n + 1);
Show[
Plot[NARFunction[x] /. #, {x, 0, 3}, PlotStyle -> {Hue[n/19 /. #], Thickness[0.01]},
DisplayFunction -> Identity] & /@ {n -> 2, n -> 4, n -> 8, n -> 15},
DisplayFunction -> $DisplayFunction, AxesLabel -> {"t", "Y[t]/K"},
ImageSize -> {200, 110}];
```



```
In[202]:=
ApproxNARFunction[x_] := 1 - UnitStep[x - 1];
Plot[ApproxNARFunction[Y[t]], {Y[t], 0, 3}, PlotStyle -> {Hue[0.9], Thickness[0.01]},
AxesLabel -> {"t", "Y[t]/K"}, ImageSize -> {200, 110}];
```



This step-function approximation is useful in deriving analytical results.



14 of 30

Motifs: Simple Regulation VS. Negative Auto-Regulation

```
In[204]:=
ApproxNAREgulationODE =
  NAREgulationODE /. {1 / (1 + (Y[t] / K) ^ n) -> ApproxNARFunction[Y[t] / K]};
% // TraditionalForm
```

```
Out[205]//TraditionalForm=

$$Y'(t) = \beta \theta(X) \left( 1 - \theta \left( \frac{Y(t)}{K} - 1 \right) \right) - \alpha Y(t)$$

```

Now lets look at the steady-state solution, further assuming that $K \ll \beta/\alpha$. Lets take the limit $\alpha \rightarrow 0$.

```
In[206]:=
XInput = X -> 1;
NARODEIC = {ApproxNAREgulationODE /. XInput};
NARSteadyStateSoln =
  Reduce[{{NARODEIC[[1]]} /. {Derivative[n_][Y_][t_] -> 0, Y[t] -> Y, \alpha -> 0}},
  {K > 0, Y > 0, \beta > 0}, Reals];
% // TraditionalForm
```

```
Out[209]//TraditionalForm=

$$Y > 0 \wedge 0 < K \leq Y \wedge \beta > 0$$

```

So, in this limit the steady-state is given by the repression coefficient, K.

```
In[720]:=
NARSteadyState = Y[Infinity] -> K
```

```
Out[720]=

$$Y[\infty] \rightarrow K$$

```



Motifs: Simple Regulation VS. Negative Auto-Regulation

```
In[398]:=
NAREgulationODE // TraditionalForm
```

```
Out[398]//TraditionalForm=

$$Y'(t) = \frac{\beta \theta(X)}{\left(\frac{Y(t)}{K}\right)^n + 1} - \alpha Y(t)$$

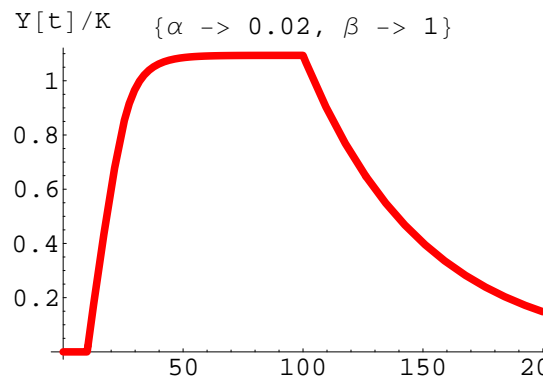
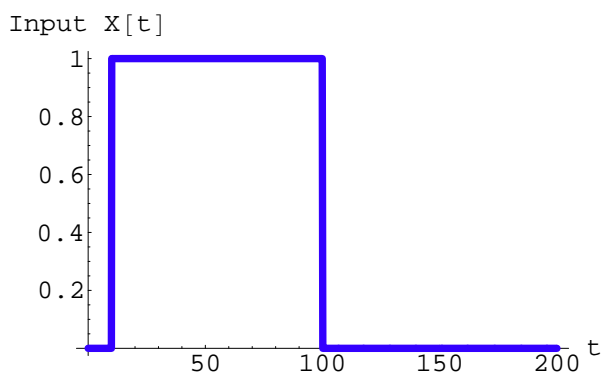
```

Lets look at the behaviors of the solution, as the value of β is varied.

```

In[447]:=
AlphaRule =  $\alpha \rightarrow 0.02$ ;
BetaRule =  $\beta \rightarrow 1$ ;
KRule =  $K \rightarrow 15$ ;
StepInputSoln =
  NDSolve[{NAREgulationODE, Y[0] == 0} /. n  $\rightarrow$  8 /. {UnitStep[X]  $\rightarrow$  StepInput} /. AlphaRule /.
    BetaRule /. KRule, Y[t], {t, 0, 200}, MaxSteps  $\rightarrow$  10^6 // Simplify // Flatten;
RespPlot[AlphaRule_, BetaRule_, KRule_] := Plot[
  Y[t] / K /. n  $\rightarrow$  8 /. KRule /. StepInputSoln /. AlphaRule /. BetaRule // Simplify,
  {t, 0, 200}, PlotStyle  $\rightarrow$  {Hue[0.0]}, Thickness[0.015],
  ImageSize  $\rightarrow$  {300, 130}, PlotRange  $\rightarrow$  All, AxesLabel  $\rightarrow$  {"t", "Y[t]/K"},
  PlotLabel  $\rightarrow$  ToString[{AlphaRule, BetaRule}], DisplayFunction  $\rightarrow$  Identity];
Show[GraphicsArray[{{StepInputPlot}, {RespPlot[AlphaRule, BetaRule, KRule]}} //
  Transpose], ImageSize  $\rightarrow$  {500, 160}];

```



16 of 30

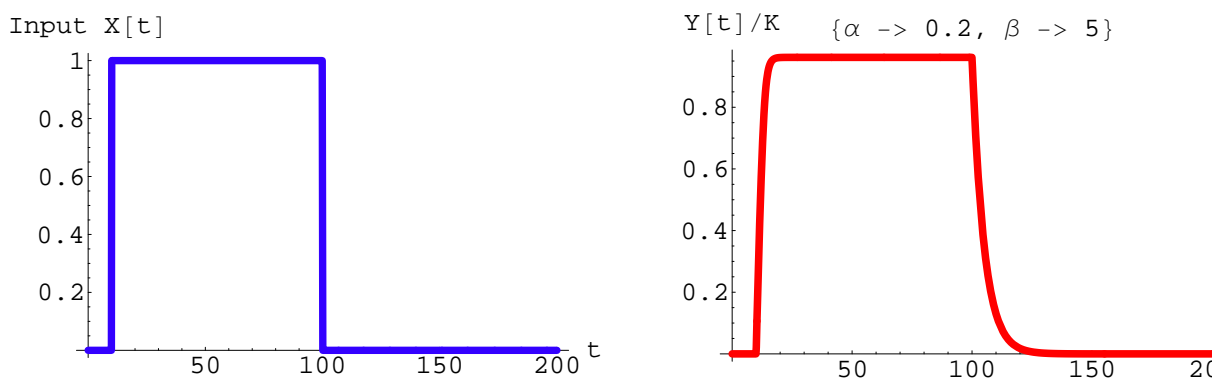
Motifs: Simple Regulation VS. Negative Auto-Regulation

Now increase β by a factor of 5.

```

In[453]:=
AlphaRule =  $\alpha \rightarrow 0.2$ ;
BetaRule =  $\beta \rightarrow 5$ ;
KRule =  $K \rightarrow 15$ ;
StepInputSoln =
  NDSolve[{NAREgulationODE, Y[0] == 0} /. n  $\rightarrow$  8 /. {UnitStep[X]  $\rightarrow$  StepInput} /. AlphaRule /.
    BetaRule /. KRule, Y[t], {t, 0, 200}, MaxSteps  $\rightarrow$  106 // Simplify // Flatten;
RespPlot[AlphaRule_, BetaRule_, KRule_] := Plot[
  Y[t] / K /. n  $\rightarrow$  8 /. KRule /. StepInputSoln /. AlphaRule /. BetaRule // Simplify,
  {t, 0, 200}, PlotStyle  $\rightarrow$  {Hue[0.0], Thickness[0.015]},
  ImageSize  $\rightarrow$  {300, 130}, PlotRange  $\rightarrow$  All, AxesLabel  $\rightarrow$  {"t", "Y[t]/K"},
  PlotLabel  $\rightarrow$  ToString[{AlphaRule, BetaRule}], DisplayFunction  $\rightarrow$  Identity];
Show[GraphicsArray[{{StepInputPlot}, {RespPlot[AlphaRule, BetaRule, KRule]}} //
  Transpose], ImageSize  $\rightarrow$  {500, 160}];

```



That is, the response is **significantly faster** with larger β .



Motifs: Simple Regulation VS. Negative Auto-Regulation

We can also understand the effect of β by simple analysis.

For initial times,

$$Y[t] \sim \beta t$$

i.e., when $Y[t] < K \ll \beta/\alpha$, the time to reach half of the steady state level is:

```

In[819]:=
HalfSStimeEqn =  $\beta t == 1/2 * Y[\infty] /. NARSteadyState$ 

```

```

Out[819]=
t  $\beta == \frac{K}{2}$ 

```

```

In[820]:=
NARHalfTime = Solve[%, t] // Flatten

```

```

Out[820]=
{t  $\rightarrow \frac{K}{2\beta}$ }

```

- That is, the stronger the maximal unrepresed promoter activity β , the shorter the response time.
- Stronger promoter (β) to give rise to initial fast production, then use auto-repression (K) to stop at the desired steady-state.

- Promoter activity β can be tuned by mutations in the DNA binding site of RNA-polymerase.
- Auto-repression constant (K) can be modified by mutations in the DNA binding site of Y in the promoter.



18 of 30

Motifs: Simple Regulation VS. Negative Auto-Regulation

Mathematically Controlled Comparison

- 2 designs: simple regulation, negative auto-regulation
- How to make mathematically controlled comparison?
 - for alternative designs:
 - maintain *internal equivalence* (e.g., values of biochemical parameters)
 - impose *external equivalence* (e.g., same final steady-states)
- Internal equivalence: choose same protein degradation rate, α
- External equivalence: choose same steady-state values
 - usually important for maintaining the optimal function

In[796]:=

```
ExternalEquivCond =
  (NARSteadyState /.  $\beta \rightarrow \beta\_NAR$ ) == (SteadyStateSoln[[1]] /. { $\beta \rightarrow \beta\_Simple$ ,  $\alpha \rightarrow \alpha\_Simple$ })
```

Out[796]=

$$(Y[\infty] \rightarrow K) == \left(Y[\infty] \rightarrow \frac{\beta_Simple}{\alpha_Simple} \right)$$

In[797]:=

```
ExternalEquivCond /. {Rule[x_, y_] -> y}
```

Out[797]=

$$K == \frac{\beta_Simple}{\alpha_Simple}$$

In[798]:=

```
ExternalEquivSoln = Solve[%, K] // Flatten
```

Out[798]=

$$\left\{ K \rightarrow \frac{\beta_Simple}{\alpha_Simple} \right\}$$



19 of 30

Motifs: Simple Regulation VS. Negative Auto-Regulation

Remember, the response time for Negative Auto-Regulation is:

```
In[821]:=
  NARHalfTime
```

```
Out[821]=
  {t ->  $\frac{K}{2\beta}$ }
```

With K chosen to maintain external equivalence,

```
In[822]:=
  NARHalfTimeNew = NARHalfTime /. ExternalEquivSoln
```

```
Out[822]=
  {t ->  $\frac{\beta\_Simple}{2\alpha\_Simple\beta}$ }
```

For simply-regulated, the response time is:

```
In[826]:=
  SRHalfTime = {t ->  $\frac{\text{Log}[2]}{\alpha}$ } /. {alpha -> alpha_Simple};
```

Now taking the ratio of the response times:

```
In[828]:=
  RatioHalfTimes = (NARHalfTimeNew / SRHalfTime) /. {Rule[x_, y_] -> y} // First // Simplify
```

```
Out[828]=
   $\frac{\beta\_Simple}{\beta \text{Log}[4]}$ 
```

20 of 30

Motifs: Simple Regulation VS. Negative Auto-Regulation

Thus, even though both internal and external equivalences are maintained, negative autoregulation allows for speeding up the response time:

```
In[829]:=
  RatioHalfTimes
```

```
Out[829]=
   $\frac{\beta\_Simple}{\beta \text{Log}[4]}$ 
```

That is, the above ratio can be made **small** by choosing β **large** compared to β_Simple .

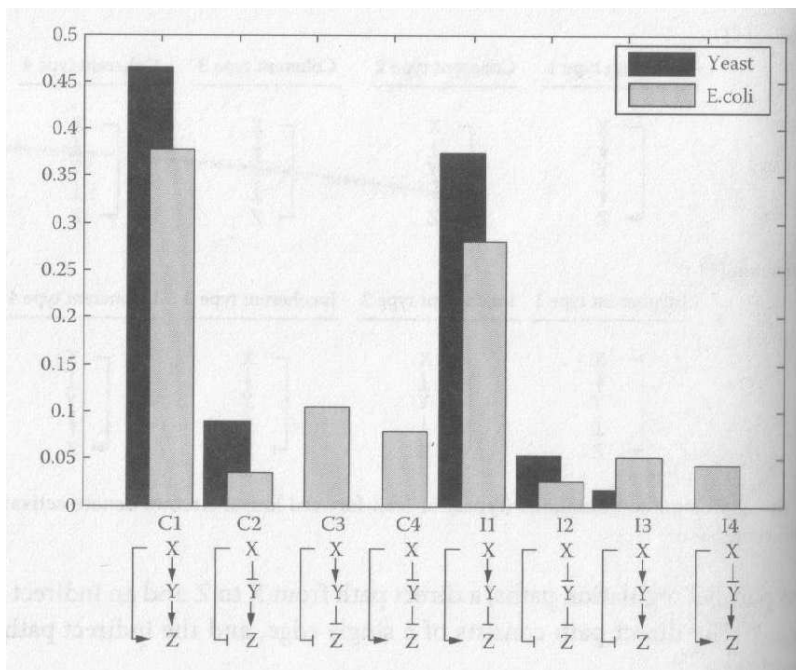
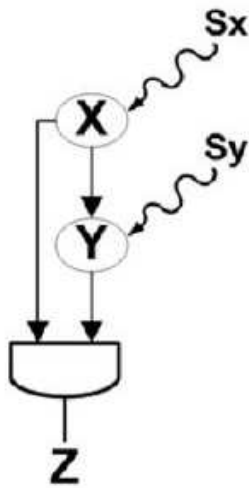
To summarize, negative auto-regulation gains by: rapid production, balanced by suitable repression as the desired steady-state is reached.

The same strong promoter on simple-regulation circuit would reach a much higher steady-state, thereby leading to undesirable overexpression of gene product.

21 of 30

Motifs: Feed-Forward Loops

C1-Feed-Forward Loop



Motifs: Feed-Forward Loops

```
In[479]:=
ClFFL_ODE_StepInput = {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]} /.
  {X[t] -> UnitStep[t]};
% // ColumnForm // TraditionalForm

Out[480]//TraditionalForm=

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


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


In[481]:=
ClFFLSoln = Simplify[DSolve[ClFFL_ODE_StepInput] // Flatten, {Y, Z}, t];
% // PiecewiseExpand

Out[482]=
{ {Y ->
  Function[{t},  $e^{-t \alpha_Y} C[1] + e^{-t \alpha_Y} \beta_Y \left( \left\{ \frac{e^{t \alpha_Y}}{\alpha_Y} \quad (K_{XY} \leq 1 \ \&\& \ t \geq 0) \mid\mid (K_{XY} \leq 0 \ \&\& \ t < 0) \right\} \right)$ ],
  Z -> Function[{t},  $e^{-t \alpha_Z} C[2] + e^{-t \alpha_Z}$ 
  ( {  $\frac{e^{t \alpha_Z} \beta_Z}{\alpha_Z} \quad (K_{XY} > 1 \ \&\& \ t \geq 0 \ \&\& \ -K_{YZ} + e^{-t \alpha_Y} C[1] \geq 0 \ \&\& \ K_{XZ} \leq 1) \mid\mid (K_{XY} \leq 1 \ \&\& \ t \geq 0)$ 
  ) } ] } }
```

23 of 30

Motifs: Feed-Forward Loops

```
In[4]:= XInput = {X[t] -> UnitStep[t]};
ClFFL_ODE_StepInput = {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[6]//TraditionalForm=

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

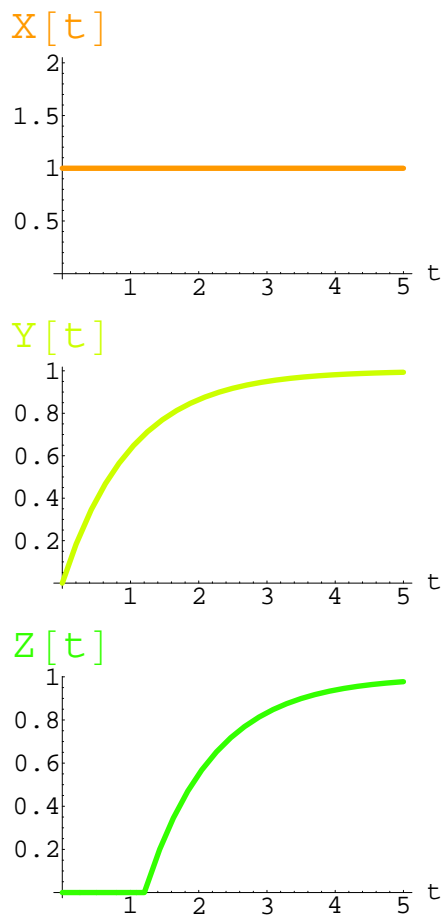

$$Z'(t) = \beta_Z \theta(\theta(t) - K_{XZ}) \theta(Y(t) - K_{YZ}) - \alpha_Z Z(t)$$


In[10]:= tEnd = 5;
ConstantsNDSolve =
  { $\alpha_Y$  -> 1,  $\alpha_Z$  -> 1,  $\beta_Y$  -> 1,  $\beta_Z$  -> 1,  $K_{YZ}$  -> 0.7,  $K_{XZ}$  -> 0,  $K_{XY}$  -> 0};
ClFFLNumSoln = ({X[t] /. XInput, Y[t], Z[t]} /. NDSolve[ClFFL_ODE_StepInput /. %,
  Y[0] == 0, Z[0] == 0, {Y[t], Z[t]}, {t, 0, tEnd}]) // Flatten;
```

24 of 30

Motifs: Feed-Forward Loops

```
In[13]:= PLabels = {"X[t]", "Y[t]", "Z[t]"};
SolnPlots = MapIndexed[Plot[#1, {t, 0, tEnd},
  ImageSize -> {200, 110}, 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] &, ClFFLNumSoln];
```

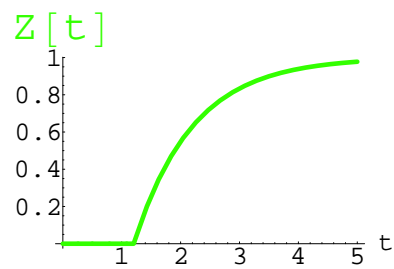
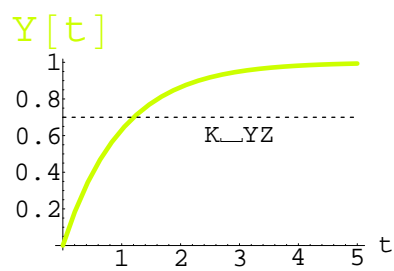
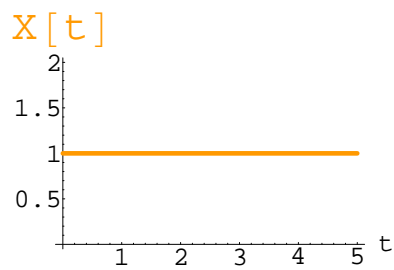


25 of 30

Motifs: Feed-Forward Loops

```
In[69]:= 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", {3, -0.1 + K_YZ /. ConstantsNDSolve}]]],
  DisplayFunction -> Identity];
```

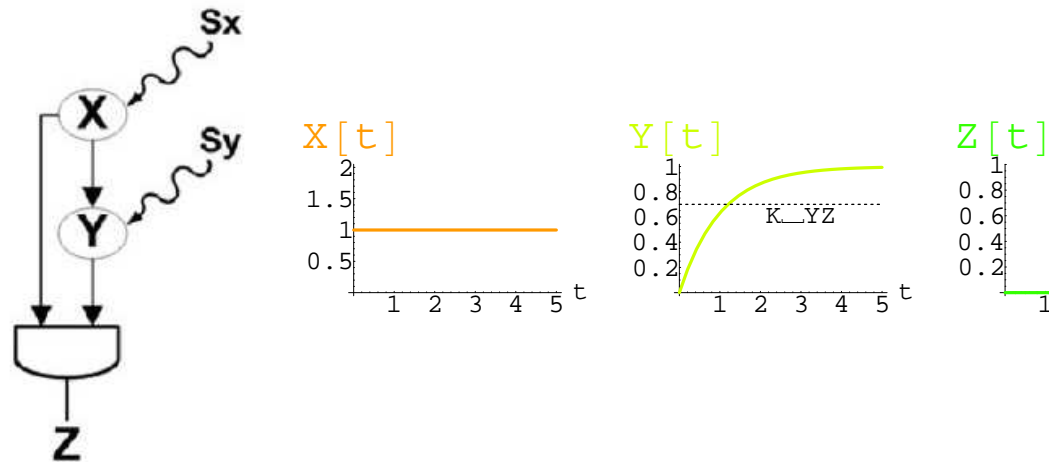
```
In[80]:= Show[GraphicsArray[{SolnPlotsNew} // Transpose],
  ImageSize -> {200, 110 * 3}, DisplayFunction -> $DisplayFunction];
```



Motifs: Feed-Forward Loops

Thus, we have shown that for an **UnitStep** increase in the input (from 0 to 1), gene Z shows a delay.

```
In[86]:= Show[GraphicsArray[{C1FFLPlot, SolnPlotsNew} // Flatten], ImageSize -> {600, 200};
```



Does the same phenomenon occur for a **UnitStep** decrease? i.e., **X** goes from 1 to 0.



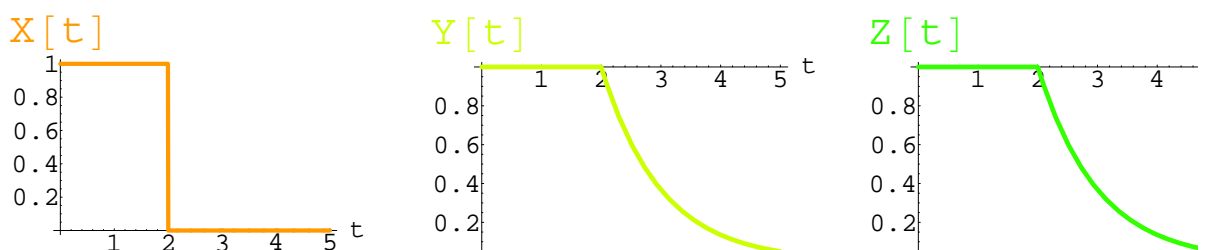
Motifs: Feed-Forward Loops

Lets try a different input: an initial 1 that steps to 0.

```
In[87]:= XInput = {X[t] -> 1 - UnitStep[t - 2]};
C1FFL_ODE_StepInput = {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[89]//TraditionalForm=
Y'(t) == beta_Y theta(-K_XY - theta(t - 2) + 1) - alpha_Y Y(t)
Z'(t) == beta_Z theta(-K_XZ - theta(t - 2) + 1) theta(Y(t) - K_YZ) - alpha_Z Z(t)
```

```
In[119]:= SolnPlots = MapIndexed[Plot[#1, {t, 0, tEnd}], ImageSize -> {200, 110},
  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] &, C1FFLNumSoln];
Show[GraphicsArray[{SolnPlots}], ImageSize -> {500, 110},
  DisplayFunction -> $DisplayFunction];
```



We see that there's **no delay** for this input.

That is, the switching is **asymmetric** with respect to increase or decrease of input.



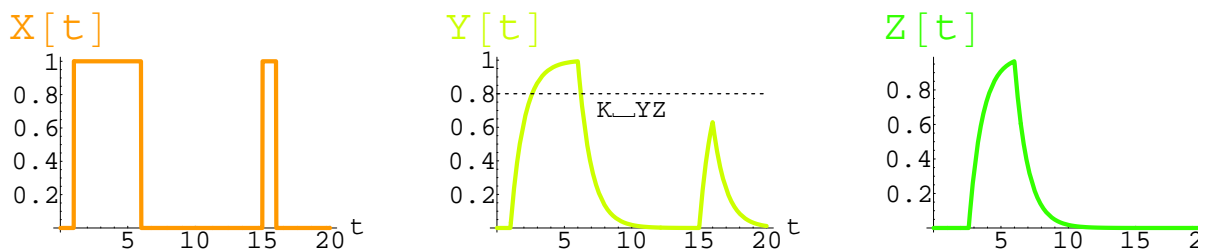
Motifs: Feed-Forward Loops

`In[181]:=`

```
XInput = {X[t] -> UnitStep[t - 1] - UnitStep[t - 6] + UnitStep[t - 15] - UnitStep[t - 16]};
C1FFL_ODE_StepInput = {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]} /. %;
```

`In[190]:=`

```
tEnd = 20;
ConstantsNDSolve = { $\alpha_Y$  -> 1,  $\alpha_Z$  -> 1,  $\beta_Y$  -> 1,  $\beta_Z$  -> 1,  $K_{YZ}$  -> 0.8,  $K_{XZ}$  -> 0.1,  $K_{XY}$  -> 0.1};
C1FFLNumSoln = ({X[t] /. XInput, Y[t], Z[t]} /.
  NDSolve[{C1FFL_ODE_StepInput /. %, Y[0] == 0, Z[0] == 0}, {Y[t], Z[t]}, {t, 0, tEnd}]) // Flatten;
SolnPlots = MapIndexed[Plot[#1, {t, 0, tEnd}], ImageSize -> {200, 110},
  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] &, C1FFLNumSoln];
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}], ImageSize -> {500, 110},
  DisplayFunction -> $DisplayFunction];
```



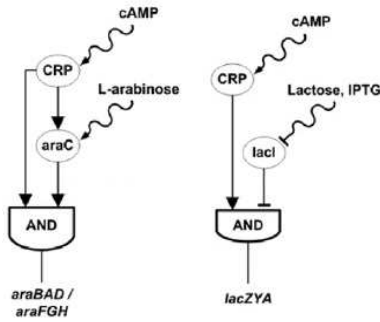
That is, the C1 Feed-Forward Loop is able to act as a signal filter: only persistent signals $X[t]$ is registered on $Z[t]$; brief signals do not switch on gene Z .



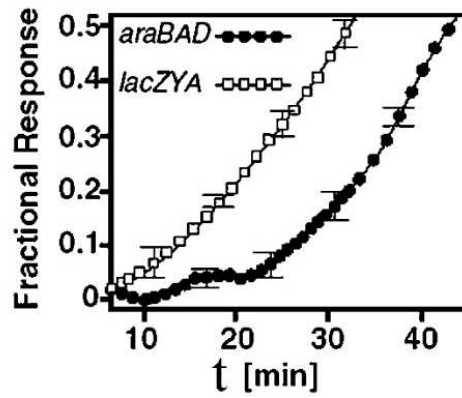
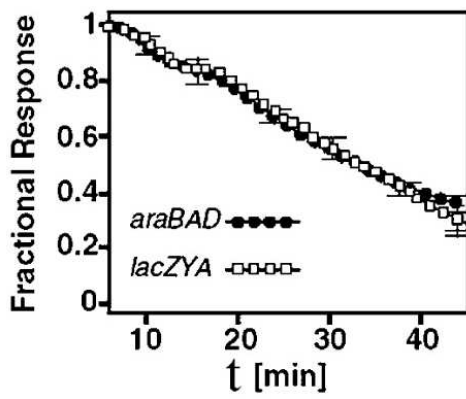
Motifs: Feed-Forward Loops

Experimental demonstration: S. Mangan, A. Zaslaver and U. Alon, J. Mol. Biol. (2003).

- 2 circuits were compared (araBAD and lacZYA systems)



– measurements of step-like response to cAMP stimuli



Conclusions:

- Different motifs are observed in different biological networks (e.g., sensory, transcriptional, neuronal)
- **Negative auto-regulation** has an advantage over simple regulation by speeding up response.
- **C1-FFL** can give act to filter brief (noisy) signals, and respond only to persistent ones.

- Next time:
 - I1-FFL (another predominant FFL motif)
 - Motifs for **temporal programs** for different orders of activation/inactivation of genes