Intermittency model description and examples

This livescript contains the following information:

- Descriptions of model parameters
- Descriptions of model equations
- Examples of how to run the model and plot results

Additional description and contextualization of the model can be found in the preprint: Tilsen, S. *Phrasal rhythmicity and the sources of temporal intermittency in speech*, and in Tilsen, S. (2019) *Syntax with oscillators and energy levels*

Please note that this model is a work-in-progress, and a number of future improvements are planned.

Table of Contents

Model parameters

The following are the parameters used in `intermittency_model.m`. The defaults are defined in the function `model_default_params.m`, but can be replaced by passing them as fields in an input structure (see how to run the model below). The variable names, default values, and short descriptions of parameters are shown below. The effects of their values can be better understood by examining the model equations. Note the following abbreviations:

- C,c : concept system
- S,s : syntactic system
- F : feedback system
- E : excitation potential
- An : annealer
- Me : excitation monitoring system

```
P = model_default_params;
dp = @(ptype)disp_pars(P,ptype);
```

Simulation parameters:

```
dp('simulation');
```

name	val	desc

```
    "T"                          5    "maximal run time"
    "stop_on_completion"     0.001    "stop if the sequence is completed"
    "dt"                     0.001    "time step"
```

Systems parameters:

```
p = dp('systems');
```

```
       name                  val                                            desc
    __________    ________________________    ______________________________________________________________________

    "Nc"          {[                 100]}    "number of c-systems"
    "Ns"          {[                   3]}    "number of s-systems"
    "Sx0"         {[0.5000 0.3000 0.1000]}    "initial syntactic system activation"
    "S_S"         {3×3 double            }    "s-to-s coupling matrix"
    "C_S_prop"    {[              0.3333]}    "proportion of systems coupled to verbal S-system"
    "Ne"          {[                   5]}    "number of initial energy levels (usually Ns + 2 extra for ground and
    "f"           {[                   8]}    "oscillator intrinsic frequency (Hz)"
```

This is the syntactic system-to-syntactic system coupling matrix for an SVO utterance, where the rows/cols correspond to N+ (S), V, and N- (O) s-systems:

```
disp(p.val{p.name=='S_S'})
```

```
     0     1     0
     1     0    -1
     0    -1     0
```

For further detail, see Tilsen (2019). *Syntax with oscillators and energy levels*.

Noise amplitudes:

```
dp('noise amplitude');
```

```
      name       val                  desc
    _________    ___    _________________________________

    "eta_Cp"     0.1    "c-system phase noise"
    "eta_Cx"     0.1    "c-system activation noise"
    "eta_Sp"     0.1    "s-system phase noise"
    "eta_Sx"     0.5    "s-system activation  noise"
    "eta_Sx0"    0.1    "s-system activation reset noise"
```

Activation coupling strength parameters. Currently the default model includes only c-to-s activation coupling.

```
dp('activation coupling strength');
```

```
      name      val          desc
    ________    ___    ________________

    "chi_SC"     0     "s-to-c coupling"
    "chi_SS"     0     "s-to-s coupling"
    "chi_CC"     0     "c-to-c coupling"
    "chi_CS"     4     "c-to-s coupling"
```

Phase coupling strength parameters:

```
dp('phase coupling strength');
```

name	val	desc
"psi_SC"	10	"s-to-c coupling"
"psi_SS"	10	"s-to-s coupling"
"psi_CC"	0	"c-to-c coupling"
"psi_CS"	0.05	"c-to-s coupling"

Global excitation potential parameters:

```
dp('global excitation potential');
```

name	val	desc
"gw_ub"	1.1	"upper limit"
"gw_lb"	0	"lower limit"
"gain_gw"	0.1	"gain"

Syntactic-system excitation potential parameters:

```
dp('s-system excitation potential');
```

name	val	desc
"sigma_VE"	0.05	"width of force regions"
"gain_VE"	1	"strength of excitation potential"

Annealing parameters:

```
dp('annealer');
```

name	val	desc
"gain_Ai"	1.2	"gain of annealer in initial organization phase"
"gain_Ar"	6	"gain of annealer after initial organization phase"

Excitation monitor parameters:

```
dp('excitation monitor');
```

name	val	desc
"Me_unocc_cost"	15	"cost of unoccupied level"
"Me_multiocc_cost"	15	"cost of multiply occupied level"
"Me_decay"	10	"decay rate"
"sigme_VMe"	0.05	"width of excitation monitor potential"

```
"reset_Me"            0.5    "threshold for Monitor-induced reset"
"reorg_Me"           -0.5    "threshold for Monitor-induced regorganization"
```

Feedback parameters:

```
dp('feedback');
```

```
     name                  val                                  desc
 ____________    ________________________    ________________________________________________

"tau_Fx_dem"    {[0.5000 0.5000 0.5000]}    "feedback threshold for syntactic systems"
"tau_Sx_sel"    {[                    1]}    "threshold for syntactic system feedback growth"
```

Model equations

The model is implemented with a simple Euler (first-order) method. At each time step every dynamical variable is updated with an equation of the form:

$$x(t) = x(t-1) + \frac{\Delta x}{\Delta t} \cdot \delta t$$

with time indices $t = 1, 2, ..., (T/\delta t)$

For convenience in presenting the equations below, $\dot{x}$ is substituted for $\frac{\Delta x}{\Delta t}$, hence:

$$x(t) = x(t-1) + \dot{x}(t) \cdot \delta t$$

The specific update equations of the models are the following:

$\theta_i^C(t) = \theta_i^C(t-1) + \dot{\theta}_i^C \cdot \delta t$ phase for concept system i

$\theta_m^S(t) = \theta_m^S(t-1) + \dot{\theta}_m^S \cdot \delta t$ phase for syntactic system m

$x_i^C(t) = x_i^C(t-1) + \dot{x}_i^C \cdot \delta t$ activation for concept system i

$x_m^S(t) = x_m^S(t-1) + \dot{x}_m^S \cdot \delta t$ activation for syntactic system m

$x_m^F(t) = x_m^F(t-1) + \dot{x}_m^F \cdot \delta t$ feedback for syntactic system m

$E_l(t) = E_l(t-1) + \dot{E}_l \cdot \delta t$ value of excitation level l

$A(t) = A(t-1) + \dot{A} \cdot \delta t$ annealer state

$M(t) = M(t-1) + \dot{M} \cdot \delta t$ excitation monitor

For phase variables, 2π-periodicity is imposed by taking the updated value modulo 2π:

$$\theta_i^C(t) = \theta_i^C(t) \pmod{2\pi}$$

$$\theta_m^S(t) = \theta_m^S(t) \pmod{2\pi}$$

Note the following:

- Concept systems are indexed by i and j
- Syntactic systems are indexed by m and n. For convenience, the indices correspond to the order in which syntactic systems are selected.
- Excitation potential levels are indexed by l
- U is a random number from a uniform distribution on the interval (0,1)
- η parameters are the noise terms described above.

Concept system phase:

$$\dot{\theta}_i^C = \eta^{C\theta} \cdot U + \omega + \psi_{SC} \cdot \sum_m \left(-\sin(\phi_{mi}^{SC}) \cdot w_{mi}^{SC}\right) + \psi_{CC} \cdot \sum_j \left(-\sin(\phi_{ij}^{CC}) \cdot w_{ij}^{CC}\right)$$

where:

$\omega = 2\pi f$ is the intrinsic oscillator frequency in radians per second

$\phi_{mi}^{SC} = \theta_i^C - \theta_m^S$ is the phase difference between the concept system i and syntactic system m

w_{mi}^{SC} is the syntactic-conceptual system coupling matrix

$\phi_{mi}^{CC} = \theta_i^C - \theta_j^C$ is the phase difference between the concept system i and concept system j

w_{mi}^{SC} is the conceptual-conceptual system coupling matrix

Syntactic system phase:

$$\dot{\theta}_m^S = \eta^{S\theta} \cdot U + \omega + \psi_{SS} \cdot \sum_n \left(-\sin(\phi_{mn}^{SS}) \cdot w_{mn}^{SS}\right) + \psi_{CS} \cdot \sum_i \left(-\sin(\phi_{im}^{CS}) \cdot w_{im}^{CS}\right)$$

where:

$\phi_{mn}^{SS} = \theta_m^S - \theta_n^S$ is the phase difference between the syntactic system m and syntactic system n

w_{mn}^{SS} is the syntactic-syntactic system coupling matrix

$\phi_{im}^{CS} = \theta_m^S - \theta_i^C$ is the phase difference between the syntactic system m and concept system i

w_{im}^{CS} is the concept-syntactic system coupling matrix

Concept system activation:

$$\dot{x}_i^C = -\left(x_i^C - \left(\mathscr{E}_i + \eta^{Cx} \cdot U\right)\right)$$

where:

$\mathscr{E}_i$ is the driving force from the environment on concept system i

Syntactic system activation:

$$\dot{x}_m^S = \eta^{Sx} \cdot U + V_m^S + W\left(x_m^S\right) + \chi_{CS} \cdot \sum_i w_{im}^{CS} \cdot (F_m^{Sx})$$

where:

$V_m^E = \alpha_{Ve} \cdot \dfrac{An}{\delta t} \cdot \sum_l V(x_m^S, E_l)$ is the force on syntactic system m associated with excitation potential forces

α_{Ve} is the gain of the excitation potential forces

An is the current state of the annealing system

$V(x_m^S, E_l) = -(x_m^S - E_l) \cdot e^{-\frac{\left(x_m^S - E_l\right)^2}{2\sigma_{VE}^2}}$ is the excitation potential force between level l and syntactic system m.

$$W\left(x_m^S\right) = \alpha_W \cdot \left(W^{\text{lower}}(x_m^S) + W^{\text{upper}}(x_m^S)\right)$$

$$W^{\text{lower}}(x_m^S) = \frac{1}{x_m^S - W_{lower}}$$

$$W^{\text{upper}}(x_m^S) = \frac{-1}{W_{upper} - x_m^S}$$

$F_m^{Sx} = \begin{cases} 1, & \text{if } F_m^x < \tau_{dem} \\ 0, & \text{if } F_m^x \geq \tau_{dem} \end{cases}$ This enforces the condition that demoted syntactic systems are no longer coupled to concepts.

Syntactic system feedback:

$$\dot{F}_m^x = \begin{cases} 1, & \text{if } x_m^S > \tau_{sel} \\ 0, & \text{if } x_m^S \leq \tau_{sel} \end{cases}$$

This states that feedback for syntactic system m increases when system m is selected.

Excitation levels:

$\dot{E}_l = 0$ Excitation level values are constant in this version of the model.

Excitation monitoring system:

$$\dot{M} = M^{unocc} \cdot \mathscr{E}^{unocc} + M^{multiocc} \cdot \mathscr{E}^{multiocc} + M^{decay} \cdot (-1 - M)$$

where:

$\mathscr{E}^{multiocc} = \sum_{mn,m \neq n} x_m^S x_n^S \cdot V^M(x_m^S, x_n^S)$ is the cost associated with multiply-occupied levels, where:

$$V^M(x_i^S, x_j^S) = e^{\frac{-\left(x_i^S - x_j^S\right)}{2\sigma_{V^M}^2}}$$

$\mathscr{E}^{unocc} = \sum_l E_l \cdot \min_m \sqrt{(E_l - x_m^S)^2}$ is the cost associated with unoccupied levels

Annealing system:

$$\dot{A} = \begin{cases} \begin{cases} A^{init}, & \text{before any selection, or after degeneracy} \\ A^{reorg}, & \text{otherwise} \end{cases}, & A < 1 \\ 0, & A \geq 1 \end{cases}$$

This simply posits that the cooling rate is A^{init} before any systems have been selected and A^{reorg} after systems have been selected, and that the cooling stops when $A = 1$.

Excitation operations:

After the above updates, excitation operations are applied, based on various conditions described below. Note that the following logical values are calculated each time step:

$sel_m = x_m^S > 1$: indicates whether syntactic system m is currently selected

$dem_m = sel_m \ \wedge \ F_m^x \geq \tau_{dem}$: flag syntactic system m for demotion if it is selected and feedback for system m is greater than or equal to the demotion threshold

$sup_m = \neg sel_m \ \wedge \ F_m^x \geq \tau_{dem}$: indices whether syntactic m is currently supressed (it is not selected but is has feedback state $\geq \tau_{dem}$)

Condition: degeneracy before any selection

$M(t) > M^{reset} \ \wedge \ \forall_m \neg sup_m$:

$A \to 0$ the annealer is reset to 0

$M \to 0$ the excitation monitoring system is reset to 0

$x_m^S \to x_0^S + \eta^{Sx0} \cdot U$ syntactic systems are reset to their initial values with a random perturbation

$C \to 0.01 \cdot C$ concept system values are diminished

Condition: degeneracy after any selection

$M(t) > M^{reset} \;\; \wedge \;\; \exists_m \;\; sup_m$:

$A \to 0$ the annealer is reset to 0

$M \to 0$ the excitation monitoring system is reset to 0

$$x_m^S \to \begin{cases} \eta^{Sx0} \cdot U + \dfrac{m}{N_{unsel}}, & \neg sup_m \\ 0.01, & sup_m \end{cases}$$

The above states that the activations of syntactic systems that have not been supressed are reset to evenly spaced values on the interval $\left[\dfrac{1}{N_{unsel}}, 1\right]$, where N_{unsel} is the number of systems that have not yet been selected, with a random perturbation $\eta^{Sx0} \cdot U$. The activations of syntactic systems that have been supressed are set to 0.01.

$C \to 0.01 \cdot C$ concept system values are diminished

Condition: initial coherence

$M(t) < M^{reorg} \;\; \wedge \;\; \forall m \;\; \neg sel_m$: when the monitor reaches a coherence threshold and not any syntactic systems have yet been selected:

$A \to 0$ the annealer is reset to 0

$M \to 0$ the excitation monitoring system is reset to 0

$E \to r^C(E)$ the canonical reorganization operation is applied to the excitation potential levels. This operation eliminates the lowest non-ground level.

$x_m^{Sx} \to 0.01, \text{if} \;\; dem_m$ the activation of demoted syntactic systems is set to 0.01

Condition: within sequence coherence

$\exists m \;\; dem_m$: when a syntactic system reaches the demotion state

$A \to 0$ the annealer is reset to 0

$M \to 0$ the excitation monitoring system is reset to 0

$E \to r^C(E)$ the canonical reorganization operation is applied to the excitation potential levels. This operation eliminates the lowest non-ground level.

$x_m^{Sx} \to 0.01, \text{if } \; dem_m$ the activation of demoted syntactic systems is set to 0.01

Examples

How to run the model and plot the results

```
sim = intermittency_model;
```

All of the dynamical variables and parameters are in the output structure (here named `sim`):

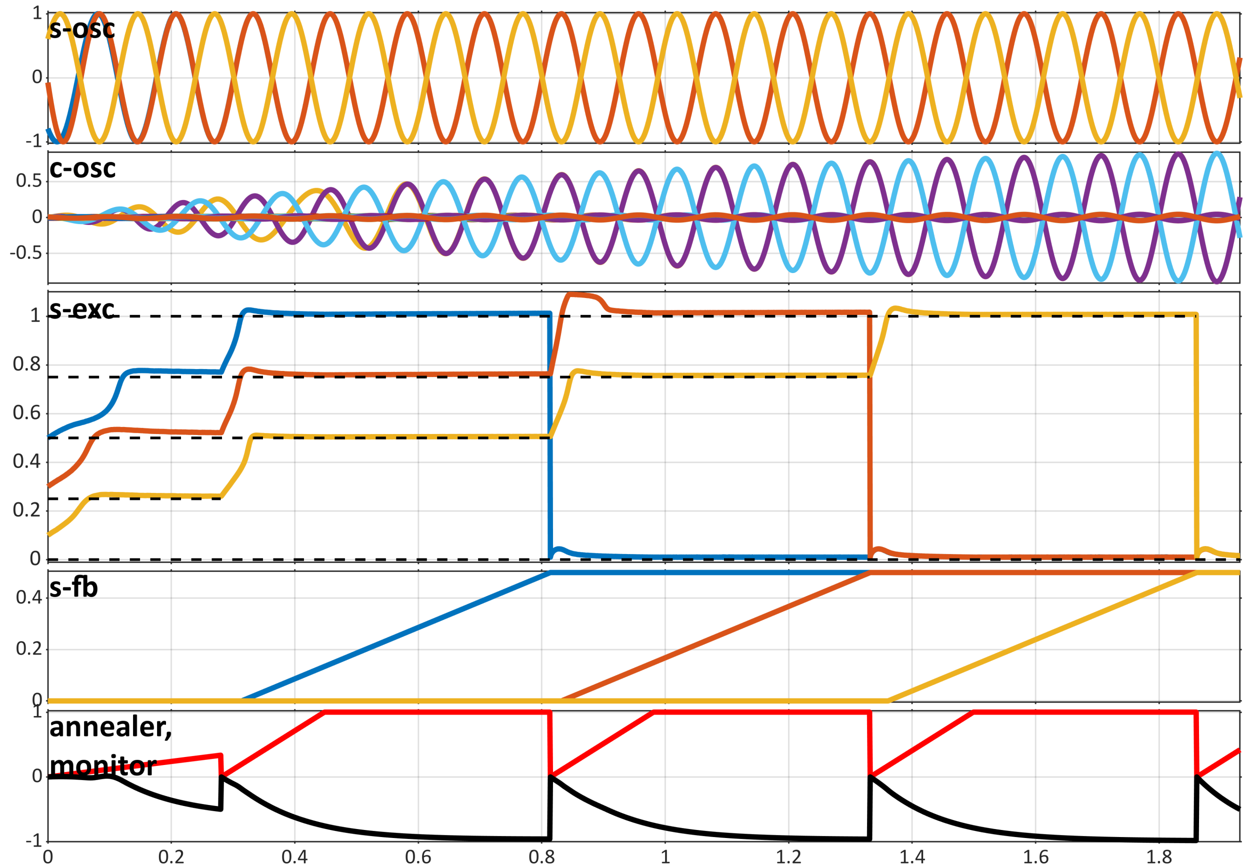
```
sim
```

```
sim = struct with fields:
                   An: [5000×1 double]
                  C_C: [100×100 double]
                  C_S: [3×100 logical]
             C_S_prop: 0.3333
                   Cp: [5000×100 double]
                   Cx: [5000×100 double]
                Cx_eq: [0.0809 0.0272 0.0325 0.0115 0.0239 0.0455 0.0426 0.0497 0.0782 1.0222 0.0577 0.0414 0.0298
                   EV: {6×2 cell}
                   Ex: [5000×5 double]
                   Fx: [5000×3 double]
                   Me: [5000×1 double]
             Me_decay: 10
     Me_multiocc_cost: 15
        Me_unocc_cost: 15
                N_ixs: [1 4 5 6 7 10 11 13 14 15 16 17 18 20 21 24 27 28 29 30 31 32 33 34 35 36 37 38 39 43 45 46
                   Nc: 100
                   Ne: 5
                   Ns: 3
                    P: [40×4 table]
               PHI_CC: [100×100 double]
               PHI_CS: [100×3 double]
               PHI_SC: [3×100 double]
               PHI_SS: [3×3 double]
                  S_S: [3×3 double]
                   Sp: [5000×3 double]
                   Sx: [5000×3 double]
                  Sx0: [0.5000 0.3000 0.1000]
                    T: 5
                 VE_S: [-6.4824 -6.3644 -6.3411]
                  V_E: @(x,c)-(x-c).*(exp(-((x-c).^2)./(2*sigma_VE^2)))
                 V_Me: @(x)exp(-(x.^2)./(2*sigme_VMe^2))
                V_ixs: [2 3 8 9 12 19 22 23 25 26 40 41 42 44 54 57 58 63 64 65 69 70 76 77 80 81 83 84 90 95]
               c_coup: [0 0 0]
               chi_CC: 0
               chi_CS: 4
               chi_SC: 0
               chi_SS: 0
                  dAn: 6
                  dCp: [50.3003 50.2887 50.3444 50.2786 50.3521 50.2786 50.3196 50.3010 50.2892 50.2838 50.3187 50.
                  dCx: [0.0366 -0.0179 -0.0117 -0.0312 -0.0192 0.0032 1.1979e-04 0.0046 0.0342 0.1236 0.0156 -0.00
                  dEx: 0
```

```
                 dFx: [0 0 0]
                 dMe: -5.0098
                 dSp: [50.2714 50.3467 50.3619]
                 dSx: [-0.0673 0.0440 0.1286]
                dgex: [0 NaN NaN NaN NaN]
                dgsx: [3×3 double]
                  dt: 1.0000e-03
           env_num_N: 2
           env_num_V: 1
              eta_Cp: 0.1000
              eta_Cx: 0.1000
              eta_Sp: 0.1000
              eta_Sx: 0.5000
             eta_Sx0: 0.1000
                   f: 8
             gain_Ai: 1.2000
             gain_Ar: 6
             gain_VE: 1
             gain_gw: 0.1000
                  gw: [5.9542 6.0800 6.1053]
               gw_lb: 0
               gw_ub: 1.1000
                   i: 1931
           init_eorg: 0
              ix_dem: [0 0 0]
        ix_diff_sign: [100×100 logical]
            ix_env_C: [0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
              ix_sel: [0 0 0]
              ix_sup: [1 1 1]
                  nT: 5000
               omega: 50.2655
              psi_CC: 0
              psi_CS: 0.0500
              psi_SC: 10
              psi_SS: 10
             rN_ixs: [10 53]
             rV_ixs: 76
            reorg_Me: -0.5000
            reset_Me: 0.5000
            sigma_VE: 0.0500
           sigme_VMe: 0.0500
  stop_on_completion: 1.0000e-03
                   t: [0 1.0000e-03 0.0020 0.0030 0.0040 0.0050 0.0060 0.0070 0.0080 0.0090 0.0100 0.0110 0.0120 0
          tau_Fx_dem: [0.5000 0.5000 0.5000]
          tau_Sx_sel: 1
                wC_S: [100×3 double]
                wS_C: [3×100 double]
```

Passing the output structure to the function `plot_sim` will plot some of the variables:

```
plot_sim(sim)
```



Changing parameters

Recall that the default parameters are specified in:

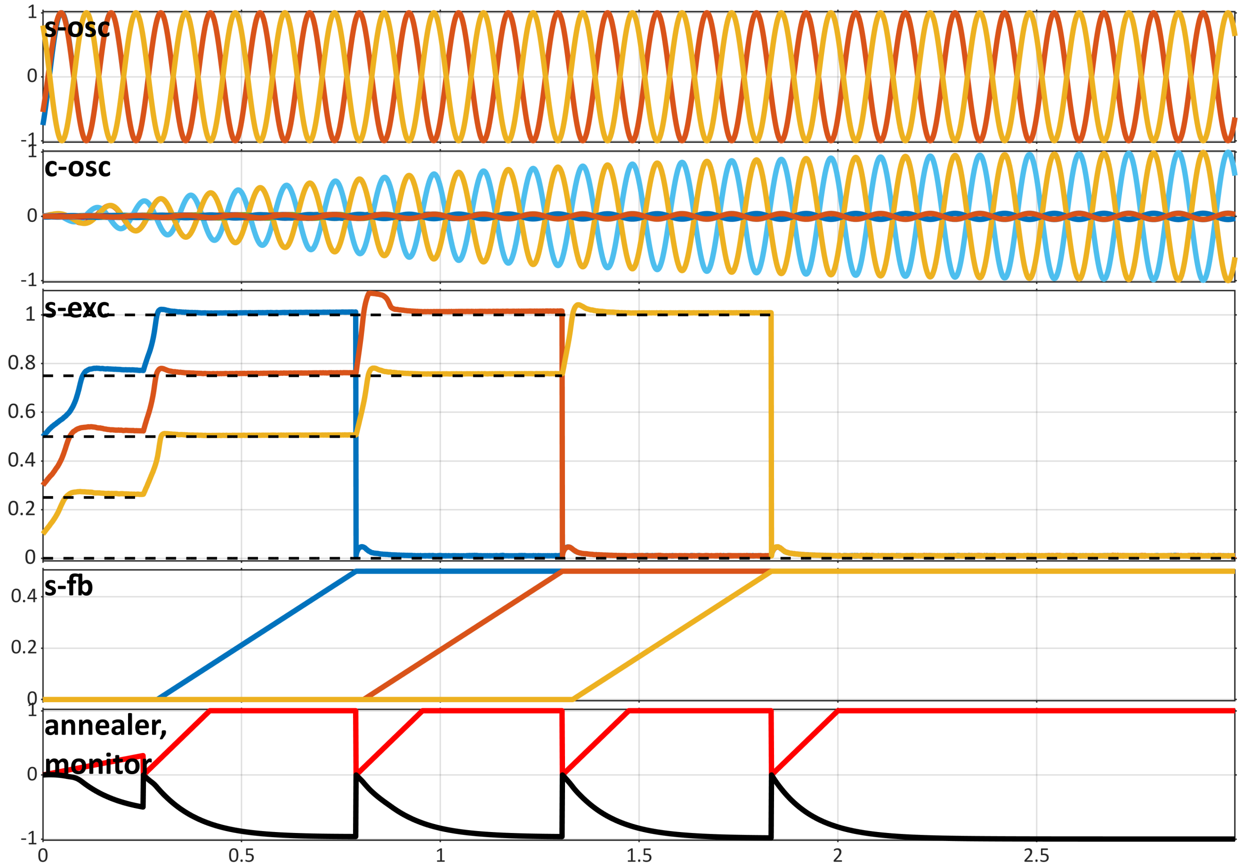
```
P = model_default_params;
head(P)
```

name	val	type	desc
{'T' }	{[5]}	{'simulation'}	{'maximal run time'
{'stop_on_completion'}	{[1.0000e-03]}	{'simulation'}	{'stop if the sequence is completed'
{'dt' }	{[1.0000e-03]}	{'simulation'}	{'time step'
{'Nc' }	{[100]}	{'systems' }	{'number of c-systems'
{'Ns' }	{[3]}	{'systems' }	{'number of s-systems'
{'Sx0' }	{[0.5000 0.3000 0.1000]}	{'systems' }	{'initial syntactic system activation'
{'S_S' }	{3×3 double }	{'systems' }	{'s-to-s coupling matrix'
{'C_S_prop' }	{[0.3333]}	{'systems' }	{'proportion of systems coupled to verba

All of the parameters can be overwritten by providing them as fields of a single input structure. For example, if we wanted to prevent the simulation from stopping upon completion of the sequence, decrease the maximal run time to 3 s, and increase the noise amplitude for syntactic system activation, we would do the following:

```
par.stop_on_completion = false;
par.T = 3;
par.eta_Sx = 1;
```

```
sim = intermittency_model(par);
plot_sim(sim)
```




```
function [P] = disp_pars(P,ptype)
P = P(ismember(P.type,ptype),{'name' 'val' 'desc'});
P.name = string(P.name);
P.desc = string(P.desc);
try
   P.val=cell2mat(P.val);
catch
end
disp(P);
end
```