Dynamic movement primitives

Gregor Schöner
gregor.schoener@ini.rub.de

Timed kinematic plans

- that have a particular time structure that emulates human movement
  - by learning from human movement data
- but also reach goals in desired time…
  - including goals that had never been encountered during movement
- => dynamic movement primitives (DMP)
  - uses a dynamical system as a basis

[Ijspeert et al., Neural Computation 25:328-373 (2013)]

Base oscillator

- damped forced harmonic oscillator

$$\tau \ddot{y} = \alpha_z(\beta_z(g - y) - \dot{y}) + f,$$

y: position

- written as two first order equations

$$\tau \dot{z} = \alpha_z(\beta_z(g - y) - z) + f,$$

$$\tau \dot{y} = z,$$

z: velocity

- has a fixed point attractor when forcing function =0

$$(z, y) = (0, g)$$

g: goal point

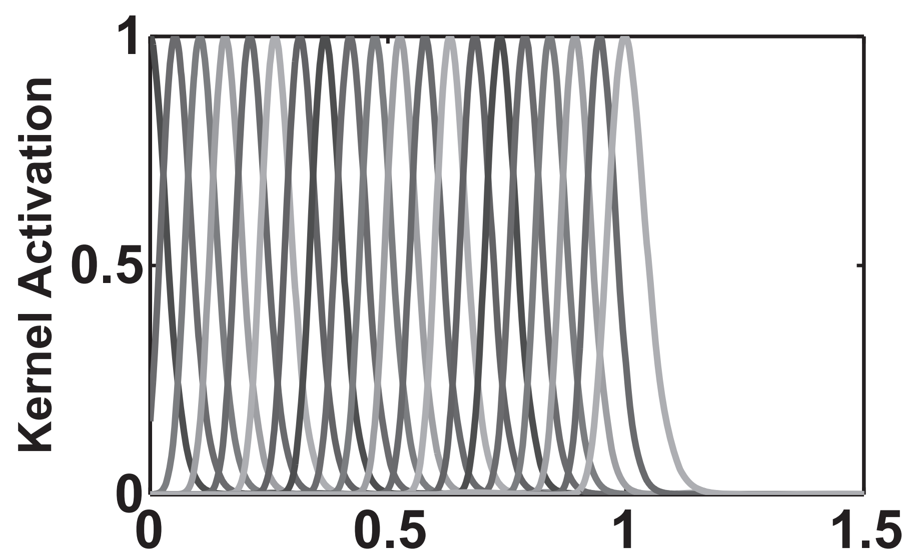
[Ijspeert et al., Neural Computation 25:328-373 (2013)]

Forcing function

- base functions
- their weighted superposition = forcing function
- explicit functions of time => non-autonomous
- staggered in time (through c_i): a time "score"

$$\Psi_i(x) = \exp\left(-\frac{1}{2\sigma_i^2}(x-c_i)^2\right),$$

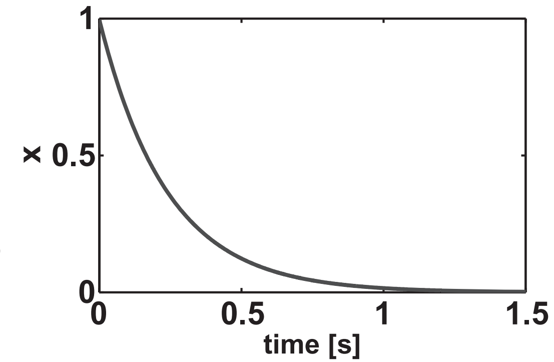
$$f(t) = \frac{\sum_{i=1}^{N} \Psi_i(t) w_i}{\sum_{i=1}^{N} \Psi_i(t)}$$



“Canonical system”

- time recoded into “phase” variable, x
- reset at each new movement initiation x(0)=1 (non-autonomous)
- scale forcing functions with amplitude and with temporal distance from end of movement
- so that forcing=0 at end of movement

$$\tau \dot{x} = -\alpha_x x,$$



$$f(x) = \frac{\sum_{i=1}^{N} \Psi_i(x) w_i}{\sum_{i=1}^{N} \Psi_i(x)} x(g - y_0)$$

y_0 initial position

$g - y_0$ amplitude

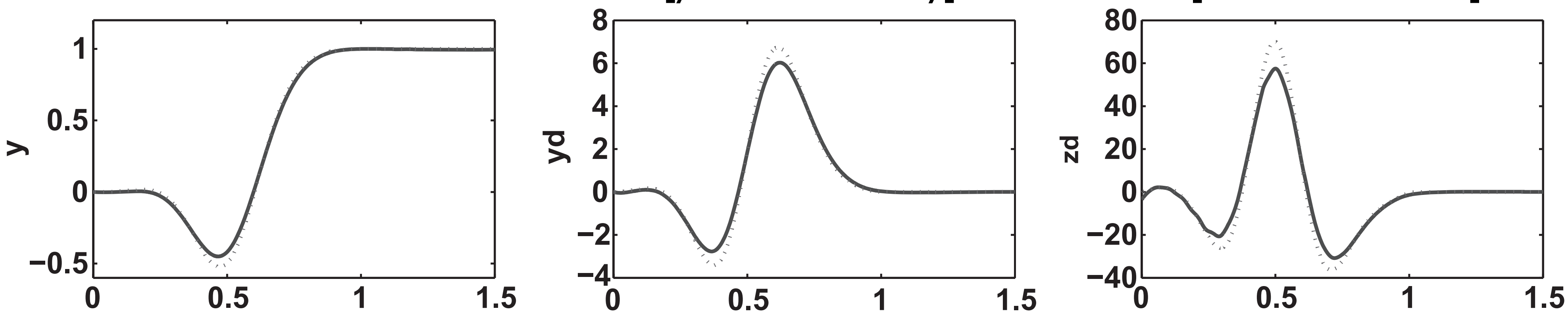
Example 1D

- weights fitted to track dotted trajectory (=5th order polynomial)… with first goes in the negative direction
- 20 kernels…

dotted: target
solid: approximation

[yd: derivative of y]

[zd: derivative of z]

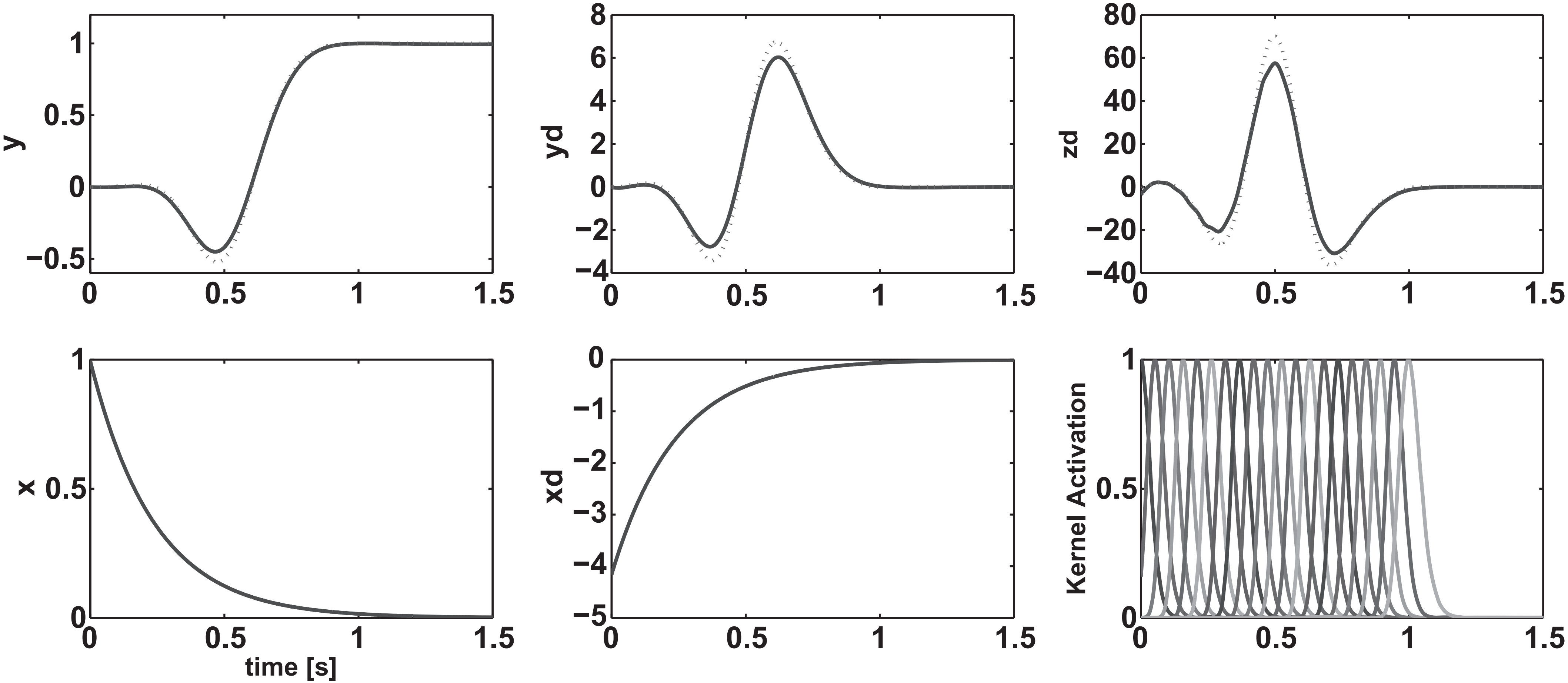


position

velocity

acceleration

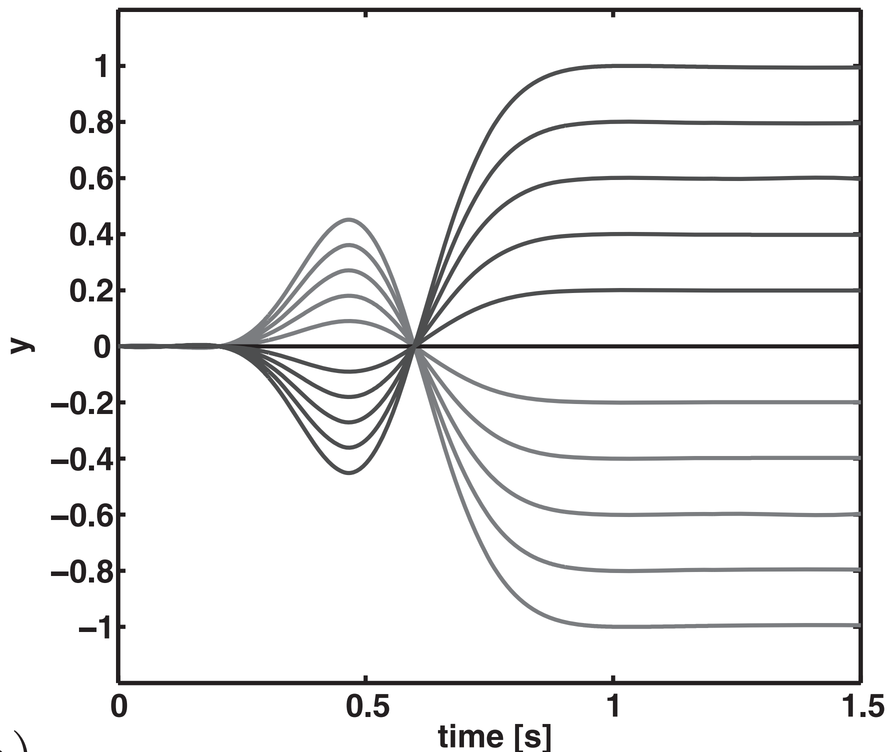
Example 1D



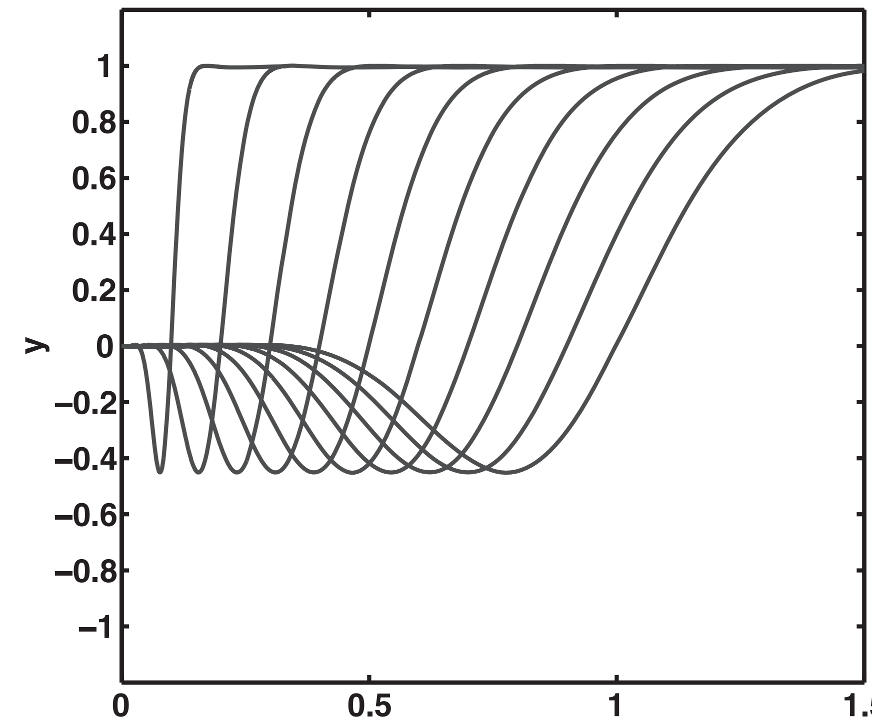
The space-time planning problem

- is to make sure the movement plan arrives at the target in a given time…
- the spatial goal is implemented by setting an attractor at the goal state
- the movement time is implicitly encoded in the tau/time scale of the "timing" variable…
  - but that relies on cutting off the timing variable, x, as some threshold level… as exponential time course never reaches zero… somewhat sensitive to that threshold…

Scaling primitives



a)



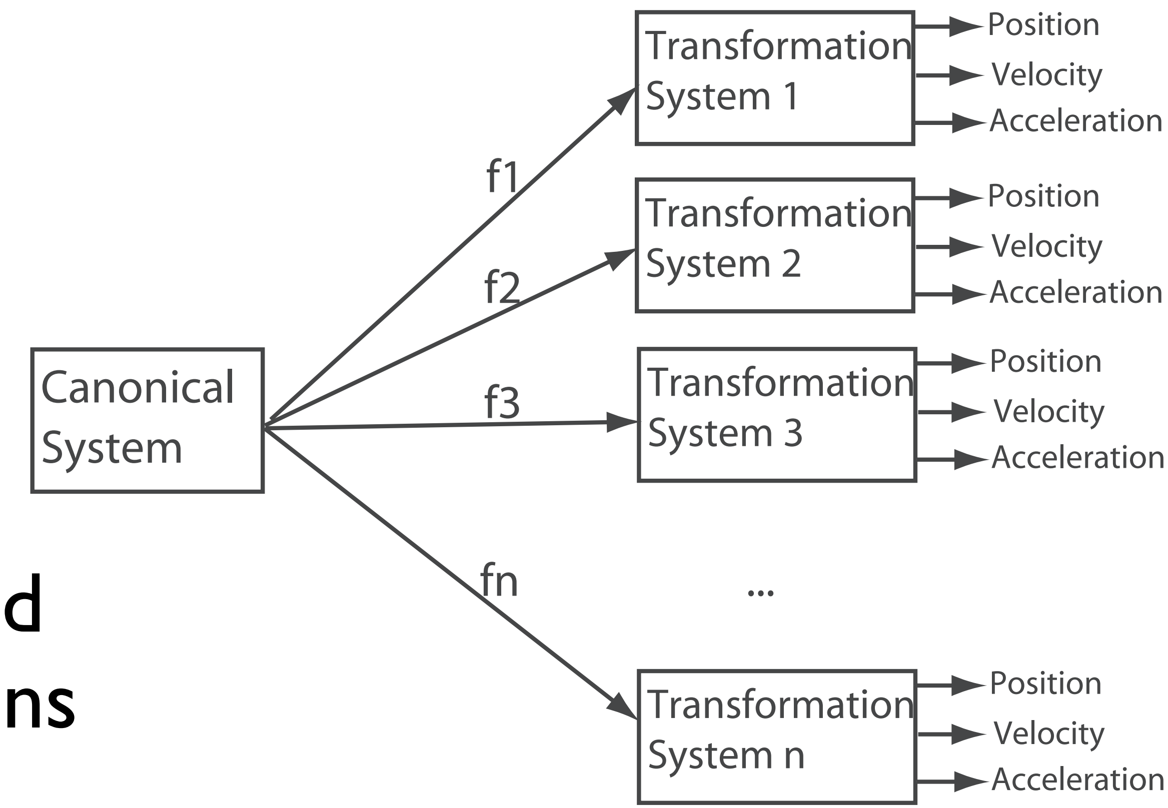
b)

scale in space from -1 to 1

scale time from 0.15 to 1.7
but: not trivially right

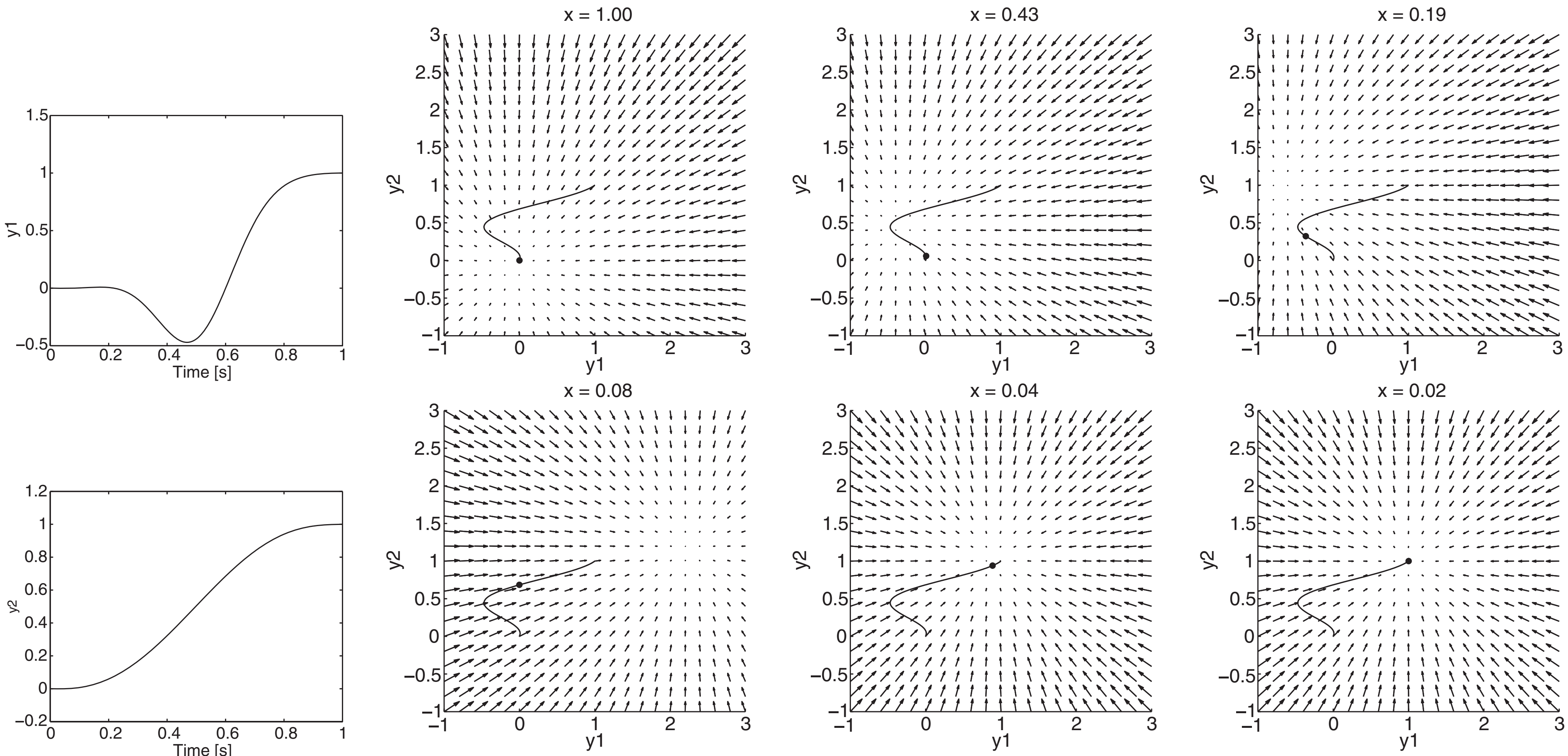
Multi-dimensional trajectories

- rather than couple multiple movement generator (deemed "complicated")…
- only one central harmonic oscillator and multiple transformations of that…



Example 2D

- single "phase" x
- two base oscillator systems y1, y2
- with two sets of forcing functions



Learning the weights

$$[\ \tau\ddot{y} = \alpha_z(\beta_z(g-y)-\dot{y})+f,\ \ ]$$

- base oscillator

$$f_{target} = \tau^2\ddot{y}_{demo} - \alpha_z(\beta_z(g-y_{demo}) - \tau\dot{y}_{demo}).$$

- forcing function from sample trajectory

$$[\quad f(x) = \frac{\sum_{i=1}^{N}\Psi_i(x)w_i}{\sum_{i=1}^{N}\Psi_i(x)}x(g-y_0)\quad]$$

- weights by minimizing error J

$$J_i = \sum_{t=1}^{P}\Psi_i(t)(f_{target}(t) - w_i\xi(t))^2,$$

$$\xi(t) = x(t)(g-y_0)$$ for discrete mov

$$\xi(t) = r$$ for rhythmic mov

Learning the weights

- can be solved analytically

minimum of

$$J_i = \sum_{t=1}^{P} \Psi_i(t)\left(f_{target}(t) - w_i \xi(t)\right)^2,$$

$$\xi(t) = x(t)(g - y_0)$$

$$\xi(t) = r$$

is

$$w_i = \frac{\mathbf{s}^T \mathbf{\Gamma}_i \mathbf{f}_{target}}{\mathbf{s}^T \mathbf{\Gamma}_i \mathbf{s}},$$

where (P=# sample times in demo trajectories):

$$\mathbf{s} = \begin{pmatrix} \xi(1) \\ \xi(2) \\ \dots \\ \xi(P) \end{pmatrix} \quad \mathbf{\Gamma}_i = \begin{pmatrix} \Psi_i(1) & & & 0 \\ & \Psi_i(2) & & \\ & & \dots & \\ 0 & & & \Psi_i(P) \end{pmatrix} \quad \mathbf{f}_{target} = \begin{pmatrix} f_{target}(1) \\ f_{target}(2) \\ \dots \\ f_{target}(P) \end{pmatrix}$$

Obstacle avoidance

- inspired by Schöner/ Dose (in Fajen Warren form)
- obstacle avoidance force-let

$$\tau \dot{z} = \alpha_z(\beta_z(g - y) - z) + f + C_t,$$

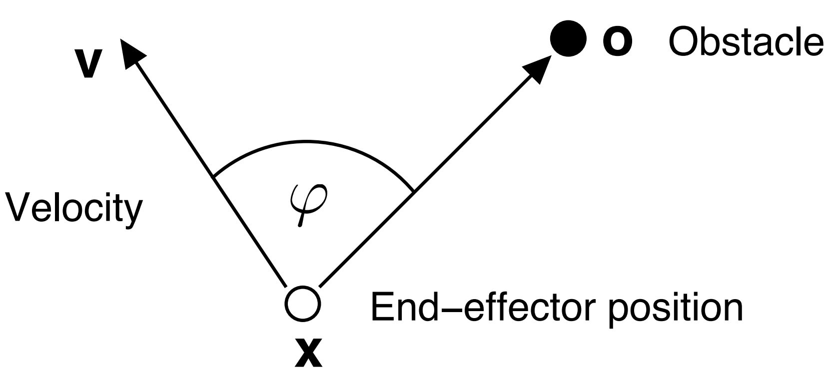
$$\tau \dot{y} = z.$$

$$\mathbf{C}_t = \gamma \mathbf{R} \dot{\mathbf{y}}\, \theta \exp(-\beta\theta),$$

where

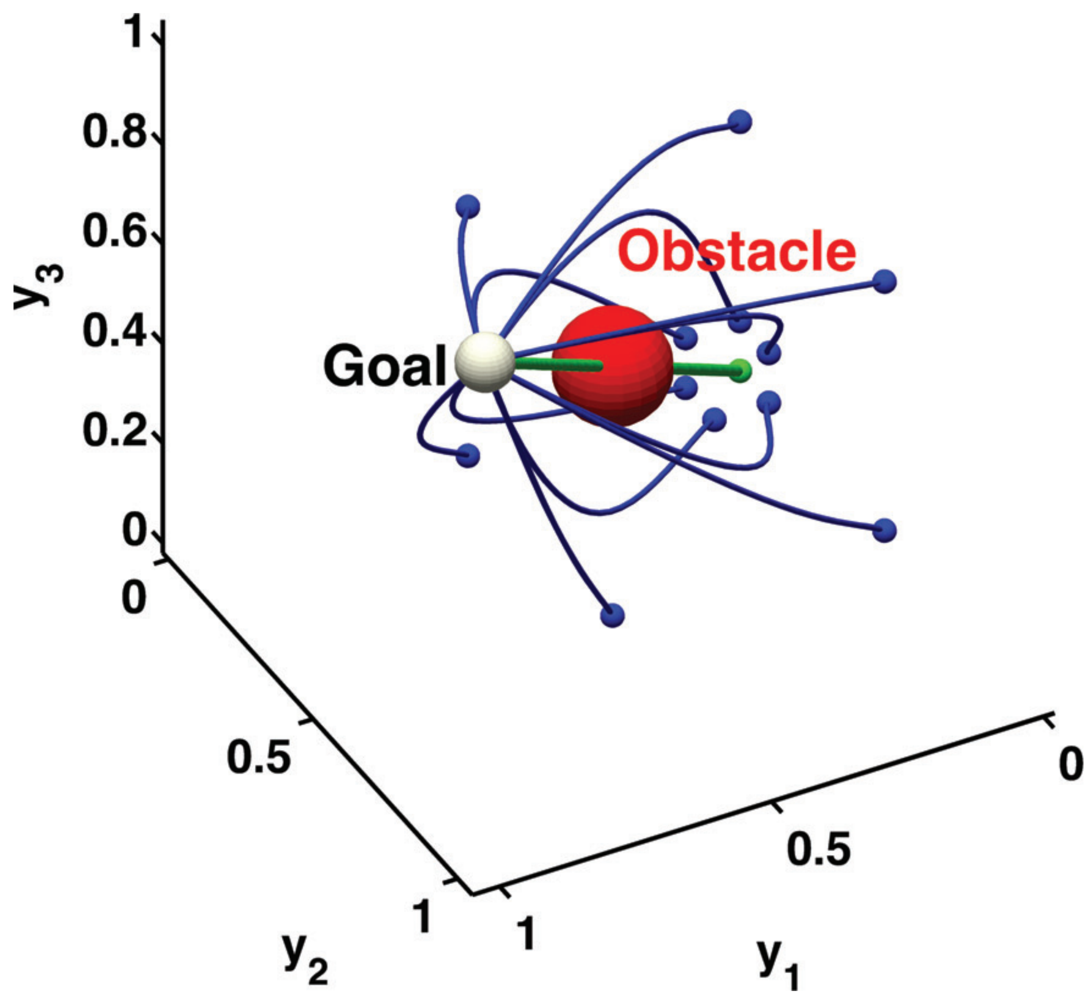
$$\theta = \arccos\left(\frac{(\mathbf{o} - \mathbf{y})^T \dot{\mathbf{y}}}{|\mathbf{o} - \mathbf{y}||\dot{\mathbf{y}}|}\right),$$

$$\mathbf{r} = (\mathbf{o} - \mathbf{y}) \times \dot{\mathbf{y}}.$$



[actually this is: Reimann, Iossifidis, Schöner, 2010]

Obstacle avoidance



But: human obstacle avoidance is not really like that…

- => Grimme, Lipinski, Schöner, 2012

Coordination

- in phase dynamics: couple to external timers…
- but: issue of predicting such events and aligning the prediction to achieve synchronicity…

$$\tau \dot{x} = -\alpha_x x + C_c$$

$$\tau \dot{\phi} = 1 + C_c.$$

$$C_c = \alpha_c (\phi_{ext} - \phi).$$

Conclusion

- DMP enable learning "movement styles" while enabling generalization to new movement targets
  - this idea can be generalized
- DMP is kinematic… control is a separate issue
  - OK for robotics, not feasible of neural control
- non-autonomous aspects of DMP are source of limitations