“Neural Engineering: Computation, Representation, and Dynamics in Neurobiological Systems”

Chris Eliasmith        Charles H. Anderson
Waterloo, Canada       St. Louis, MO

Supported by the Mathers Foundation and the McDonnell Center for Higher Brain Function.

http://compneuro.uwaterloo.ca
I also want to acknowledge my interactions with David Van Essen over the past 19 years and the support he has given me over the last 10.
Implementation

“Theorems about Turing machines and universal computation are only true in the limit of infinite resources and infinite time.”

(John Denker and Yann LeCun, Bell Labs)

Implementation issues are therefore very important !!!
The First Question To Ask Is ...
The First Question To Ask Is …

How do neuronal systems represent analog, real valued variables?
Why?
Why?

Because Neuronal Systems Process Analog Signals
Neuronal Systems are Analog

1. Sensory Inputs (Vestibular – Visual)

Vestibular Signals:

1. Angular Head Acceleration
2. Linear Head Acceleration
3. 6 Degrees of Freedom
4. ~1000 Neurons
Neuronal Systems are Analog

1. Sensory Inputs (Vestibular – Visual)
2. Motor Outputs

- ~100 Degrees of Freedom
- ~700 Muscles
- ~1,000,000 Motor Neurons
Neuronal Systems are Analog

1. Sensory Inputs (Vestibular – Visual)
2. Motor Outputs
3. Internal Cortical/Cognitive (Probabilities)
Neuronal Systems are Analog

1. Sensory Inputs (Vestibular – Visual)
2. Motor Outputs
3. Internal Cortical/Cognitive (Probabilities)

Some have suggested there are $\sim 100$ neurons per degree of freedom in cortical circuits.
So how do digital and neuronal systems represent real numbers?
Bit Representation

System Variables

A/D Converter

$x$

$\textbf{b}_n(\textbf{X})$

$\textbf{b}_n(\textbf{X})$
Bit Representation

System Variables

A/D Converter

$X$

$\hat{x} = \sum_{n} 2^n b_n(X)$

Bit Variables

$b_n(X)$
Neuronal Representation

System Variables

\[ J_i(X) = \sum_{k} \alpha_i \beta_k^X \cdot X + J_i^{\text{bias}} \]

\[ \delta[J_i(X)] \rightarrow \alpha_i(X) \rightarrow \sum_{\nu} \delta(t - \tau_{\nu}) \]

Neuron Variables

\[ \alpha_i(X) \]

\[ \sum_{\nu} \delta(t - \tau_{\nu}) \]
Neuronal Encoding

\[ J_i(X) = g_i \tilde{\phi}_i^X \cdot X + J_i^{bias} \]

\[ \mathcal{G}[J_i(X)] \rightarrow a_i(X) \]
\[ \rightarrow \sum_n \delta(t - t_i(n)) \]
Neuronal Representation

System Variables

Neuron Variables

\[
J_a(X) - g_x \phi_x^X \cdot X + J_z^{\phi_{\alpha}} \\
g_{z}J_{a}(X) \rightarrow \alpha_{z}(X) \\
\rightarrow \sum_{n} \delta(t - t_{z}(n))
\]

\[
\dot{X} - \sum_{z} \phi_{z}^{X} \alpha_{z}(X) \\
- \sum_{z, n} \phi_{z}^{X} \delta(t - t_{z}(n))
\]
Neuronal Linear Decoding

\[ \hat{X} = \sum_{i} \phi_{i}^{X} a_{i}(X) \]
\[ = \sum_{i,n} \phi_{i}^{X} h(t - t_{i}(n)) \]

Population Vector: Georgopolous et al.

Linear Temporal Filters: Bialek et al.
Optimal Linear Decoding Vectors

\[ \mathbf{X}(t) \rightarrow \mathcal{G}_i[\tilde{\phi}_i^x \cdot \mathbf{X}(t)] = \left\{ a_i(t), \text{ or } \sum_n \delta(t - t_i[n]) \right\} \]

\[ \hat{\mathbf{X}}(t) = \left\{ \sum_i \phi_i^x a_i(t), \text{ or } \sum_i \phi_i^x(t) * \sum_n \delta(t - t_i(n)) \right\} \]

\[ \text{MSEError} = \langle [\mathbf{X}(t) - \sum_i \phi_i^x a_i(t)]^2 >_{\Omega(\mathbf{X}(t))} \]

\[ \phi_i^x = \sum_j \Gamma_{ij}^{-1} V_j \]

\[ \Gamma_{ij} = \langle a_i(\mathbf{X}(t)) a_j(\mathbf{X}(t)) >_{\Omega(\mathbf{X}(t))} \]

\[ V_j = \langle a_i(\mathbf{X}(t)) \mathbf{X}(t) >_{\Omega(\mathbf{X}(t))} \]
Linear Decoding

Representation

Encoding

Decoding (Linearity)

Neuron Tuning Curves

Firing Rate vs. x

Estimate vs. Actual

x
Decoding Errors

1) Noise \sim \frac{1}{\sqrt{N}}
2) Static distortion \sim \frac{1}{N}
3) Residuals from filtered spikes \sim \frac{1}{\sqrt{N}}
Bits vs. Neurons

- Bits: (Almost infinite S/N ratio)
  1. Precision scales as $2^N$, ($N=\#\text{bits}$).
  2. Fragile and costly in power.
Bits vs. Neurons

• Bits: (Almost infinite S/N ratio)
  1. Precision scales as $2^N$, ($N=\#\text{bits}$).
  2. Fragile and costly in power.

• Neurons: (S/N Ratio $\sim 5/1$)
  1. Precision scales as $\sqrt{N}$, ($N=\#\text{Neurons}$).
  2. Fails gracefully, runs slower and/or loses precision…
Neurons work at the level of bits!!!
3 Key Elements of Framework

1) Neuronal Representation of Variables (Population codes).

2) Neural Implementation of Transformations (Synaptic coupling weights).

3) Mapping of Neuronal Dynamics onto Linear Control Theory (Spikes and Post-synaptic filters).
Neuronal Representations

- Neuronal Variables: neuron outputs
  (as instantaneous firing rates or spike trains)

\[
a_i(t), \sum_n \delta(t - t_i[n])
\]
Neuronal Representations

- **Neuronal Variables:** neuron outputs
  (as instantaneous firing rates or spike trains)

- **Higher-level Variables:** mathematical objects
  (scalars, vectors, functions, vector fields, probability functions, etc.)

### Examples

- \( a_i(t), \sum_n \delta(t - t_i[n]) \)
- **Velocity:** \( V(t) \) m/s
Neuronal Representations

- Neuronal Variables: neuron outputs
  (as instantaneous firing rates or spike trains)

- Higher-level Variables: mathematical objects
  (scalars, vectors, functions, vector fields, probability functions, etc.)

Examples

<table>
<thead>
<tr>
<th>Representation</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_i(t), \sum_n \delta(t - t_i[n])$</td>
<td>Velocity: $V(t)$ m/s</td>
</tr>
<tr>
<td>(a_i(V(t))) = (G_i[g_i &lt; \tilde{\phi}_i \cdot V(t) + J_i^{\text{bias}}])</td>
<td></td>
</tr>
</tbody>
</table>

(1) The higher-level variables are encoded into the neuronal variables using **nonlinear** spike generation rules,
Neuronal Representations

- **Neuronal Variables:** neuron outputs (as instantaneous firing rates or spike trains)

- **Higher-level Variables:** mathematical objects (scalars, vectors, functions, vector fields, probability functions, etc.)

### REPRESENTATION

1. The higher-level variables are encoded into the neuronal variables using **nonlinear** spike generation rules,

   \[ a_i(t), \sum_n \delta(t - t_i[n]) \]

2. And decoded from the neuron variables using **linear** decoding, where a weighted average is taken over neural populations and over time, i.e. linear filters.

\[
\tilde{V}(t) = \sum_i \phi_i a_i(t)
\]

\[
\hat{V}(t) = \sum_i \phi_i \sum_n h(t - t_i[n])
\]

### Examples

- **Velocity:** \( V(t) \) m/s

<table>
<thead>
<tr>
<th>( a_i(V(t)) = )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( G_i[g_i &lt; \tilde{\phi}_i \cdot V(t) &gt; + J_i^{bias}] )</td>
</tr>
</tbody>
</table>
Cochlear nerve
Vestibular nerve
Vestibulocochlear nerve (VIII)
Foramen magnum

Midsagittal line

Cochlea

Semicircular ducts:
- Anterior
- Horizontal
- Posterior

Internal acoustic meatus

Angle at which the plane of the anterior semicircular duct crosses the midsagittal line
FIGURE 33–11
The axes of polarity (arrows) of all hair cells in the macula of the utricle are oriented toward the striola, a curved border running across the surface of the macula (dotted line). Therefore, tilt in any direction depolarizes some cells and hyperpolarizes others, while having no effect on a third group. (Adapted from Spoendlin, 1966.)
Moving on to Spiking Neurons
Leaky Integrate and Fire Spiking Model

\[ \frac{dV(t)}{dt} = \frac{1}{\tau_{RC}} \left[ R \ast J(t) - V(t) \right] \]

\[ \tau_{RC} = RC. \]

If \( V(t) < 0 \) then \( V(t) = 0 \)
else if \( V(t) = V_t \) then \( V(t) = 0 \) for \( t = \tau_{ref} \).

No loss in generality setting \( V_t = 1, R = 1 \)

Reminder

\[ J_i(t) = g_i \widetilde{\phi}_i^x \cdot X(t) + J_i^{bias} \]
Firing Rate

Input voltage

Firing Rate

On

X

X

Input voltage

On
Firing Rate

Input voltage

Off

On

Firing Rate

OnOff

x

X
Firing Rate

Input voltage vs. Firing Rate. The graph shows the firing rate for different input voltages, with separate areas for 'Off', 'On', and 'Diff' conditions. The graph includes a line for 'On' firing rate, another for 'Off' firing rate, and a third for 'Diff' firing rate, which shows a deviation from the linear relationship.
Input $S(t) = A \sin(\omega t)$

Output $S^{(est)}(t) = \sum_i h(t-t_i[j]) - \sum_i h(t-t_i[j])$
Input $S(t) = A \sin(\omega t)$

$S^{(\text{on})}(t) = \sum_i h(t-t_{\text{on},i}) - \sum_j h(t-t_{\text{off},j})$
Reminder to run Matlab
On-Off Pair

2nd 100 Neurons
Linear decoding can do more than support representation.
Population Representation

- Diversity of neuronal responses is important!

**Representation**

- **Encoding**
  - Neuron Tuning Curves

- **Decoding** (Linearity)

**Neuron Tuning Curves**

**Firing Rate**

**x**

**Estimate**

**Actual**
Population Representation

- Linear decoders also support functions of x.
Neuron activities, $a(x)$
Principle Components, $\psi_n(x)$
$a_1(x)$ Along Principle Component Axes

$\psi_3(x)$

$\psi_4(x)$

$\psi_2(x)$
Computing Functions by Linear Decoding

\[ \hat{f}(X) = \sum \phi_i a_i(x) \]

\[ f(X(1), X(2)) = X(1) \ast X(2) \]
Moving on
3 Key Elements of Framework

1) Neuronal Representation of Variables (Population codes).

2) Neural Implementation of Transformations (Synaptic coupling weights).

3) Mapping of Neuronal Dynamics onto Linear Control Theory (Spikes and Post-synaptic filters).
Transformations

\[ X \rightarrow \alpha_\tau(X) \rightarrow X \]
Transformations

\[ \alpha_t(X) \rightarrow \quad Y \quad \rightarrow \quad b_j(Y) \]
Transformations

X \rightarrow Y = MX \rightarrow Y

\alpha_z(X) \rightarrow \beta_j(Y)
Transformations

\[ Y = MX \]

\[ b_j(X) = g_j \left[ \sum_{i} \omega_{ji} a_i(X) \right] \]
Transformations

\[ b_j(\mathbf{X}) = g_j \left[ \sum_i \omega_{ji} a_i(\mathbf{X}) \right] \]

\[ b_j(\mathbf{Y}) = g_j [ g_j \phi_j^{\mathbf{Y}} \cdot \mathbf{Y} + J_j^{bias} ] = g_j [ g_j \phi_j^{\mathbf{Y}} \cdot \mathbf{M} \cdot \mathbf{X} + J_j^{bias} ] = g_j [ g_j \phi_j^{\mathbf{Y}} \cdot \mathbf{M} \cdot \sum_i \phi_i^x a_i(\mathbf{X}) + J_j^{bias} ] = g_j [ g_j \sum_i \omega_{ji} a_i(\mathbf{X}) + J_j^{bias} ] \]
Neuron Coupling Weights

\[ \omega_{ji} = \tilde{\phi}_j^y M \phi_i^x \]

- Decode the incoming spikes.
- Applies the transformation.
- Encodes the result back into soma currents that generate the output spikes.
Neuron Coupling Weights

\[ \omega_{ji} = \tilde{\phi}_j^y M \phi_i^x \]

- Decode the incoming spikes.
- Applies the transformation.
- Encodes the result back into soma currents that generate the output spikes.

Learning coupling weights is complex!!
3 Key Elements of Framework

1) Neuronal Representation of Variables (Population codes).

2) Neural Implementation of Transformations (Synaptic coupling weights).

3) Mapping of Neuronal Dynamics onto Linear Control Theory (Spikes and Post-synaptic filters).
Modern Control Theory

\[ \frac{dX(t)}{dt} = AX(t) + BU(t) \]
Neural Dynamics and Control Theory

- Modern linear control theory

\[ h(s) = \frac{1}{s} \]

- Neural Dynamics

\[ h(s) = \frac{1}{1 + s \tau_{syn}} \]

\[ A' = \tau_{syn} A + 1 \]
\[ B' = \tau_{syn} B \]
Neural Dynamics

\[
\frac{dX(t)}{dt} = AX(t) + BU(t)
\]

- Filter - Integrator
- Oscillator

<table>
<thead>
<tr>
<th>A</th>
<th>A' = \tau_{syn}A + 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>\begin{bmatrix} -\frac{1}{\tau} &amp; 0 \ 0 &amp; -\frac{1}{\tau} \end{bmatrix}</td>
<td>\begin{bmatrix} 1 - \frac{\tau_{syn}}{\tau} &amp; 0 \ 0 &amp; 1 - \frac{\tau_{syn}}{\tau} \end{bmatrix}</td>
</tr>
<tr>
<td>\begin{bmatrix} 0 &amp; 1 \ -\omega^2 &amp; 0 \end{bmatrix}</td>
<td>\begin{bmatrix} 1 \ -\tau_{syn}\omega^2 \ 1 \end{bmatrix}</td>
</tr>
</tbody>
</table>
Neural Dynamics

\[
\frac{dX(t)}{dt} = AX(t) + BU(t)
\]

- Filter - Integrator
- Oscillator

Changing the coefficients in the A Matrix changes the dynamics!!

\[
A = \begin{bmatrix}
-\frac{1}{\tau} & 0 \\
0 & -\frac{1}{\tau}
\end{bmatrix}
\]

\[
A' = \tau_{syn}A + 1 = \begin{bmatrix}
1 - \frac{\tau_{syn}}{\tau} & 0 \\
0 & 1 - \frac{\tau_{syn}}{\tau}
\end{bmatrix}
\]

\[
A' = \begin{bmatrix}
0 & 1 \\
-\omega^2 & 0
\end{bmatrix}
\]

\[
A' = \begin{bmatrix}
1 & \tau_{syn} \\
-\tau_{syn}\omega^2 & 1
\end{bmatrix}
\]
Neuron Coupling Weights

\[ \omega_{ji} = \tilde{\phi}_j^y M \phi_i^x \]

- Decode the incoming spikes.
- Applies the transformation.
- Encodes the result back into soma currents that generate the output spikes.

Learning coupling weights is complex!!
Synthesis (Putting it together)

- **Decoding dynamics**
- **Matricies**
- **Encoding input spikes**
- **Output spikes**
- **Recurrent connections**
- **Neuron soma**
- **Neuronal-level description**
- **Higher-level description**
- **Dendrites**
- **Synaptic weights**

- **PSCs**

- **Input spikes**
- **Output spikes**
Synthesis (details)

Higher-level description

Decoding dynamics

Recurrent connections

Neuronal-level description

Synaptic weights

Dendrites

Encoding

Matricies

Input spikes

Output spikes

\[ \sum_{i} \delta_i^\beta(t - t_{in}) \]

\[ \sum_{m} \delta_m^\alpha(t - t_{in}) \]

\[ \frac{1}{1 + sT_0} \]

\[ \frac{1}{1 + sT_1} \]

\[ \sum_{i} \delta_i^\alpha(t - t_{in}) \]
Applications

- **Sensory Systems:**
  1. Visual System (Brandon Westover, David Van Essen)
  2. Sound Localization in Barn Owl (Brian Fischer)
  3. Vestibular System (Dora Angelaki)

- **Motor Systems:**
  1. Neural Integrator (Chris Eliasmith) --- {S. Seung}
  2. Lamprey (Chris Eliasmith)
  3. Arm movements (Zoran Nenadic)

- **Cortical Systems:**
  1. V1 (Brandon, Greg DeAngelis and David Van Essen)
  2. MT (Harris Nover and Greg DeAngelis)
  3. PDF’s (John Clark, Mike Barber)
Major Unanswered Questions

1. How to incorporate learning.
2. Efficient implementation of contextual – multiplicative interactions.
3. What is the source of the Poisson like fluctuations observed in cortical and cerebellar Purkinjie cells?
Summary: A Unified Approach to Neuroscience

• Applicable to sensory-motor-cortical systems; insects to humans.
• Links engineering and neuroscience.
• Bridges high level and low level modeling.
• A “compiler” for building model circuits.

http://compneuro.uwaterloo.ca
Neural-Simulations Matlab

http://compneuro.uwaterloo.ca

1. Gen_ensemble.m
   • Creates neuronal ensembles that represent vector spaces.

2. Setupsim.m
   • Sets up a simulation from a text file describing the circuit.

3. Runsim.m
   • Runs the simulation.

C. H. Anderson, Chris Eliasmith and John Harwell
\[
\frac{dy}{dt} = -\frac{1}{\tau_{syn}} y(t) + \frac{1}{\tau_{syn}} k(t) y(t) + x(t)
\]
\[
= -\frac{1}{\tau_{eff}(t)} y(t) + x(t)
\]
\[
\tau_{eff}(t) = \frac{\tau_{syn}}{1 - k(t)}
\]
Dynamic Filter.txt

4 Ensembles

Ensemble 1 X_N100D1 (x)
Ensemble 2 Y_N100D1 (k)
Ensemble 3 Z_N100D1 (y)
Ensemble 4 X_N100D2 (y,k)

Quad
0 1  %% Ensemble 4 will output the function k*y
0 0

Connections

Ensemble 1 (x)
  External 1

Ensemble 2 (k) Feedback gain control
  External 2

Ensemble 3 (y)
  Vector 1
  Matrix
  0.1 %% Scale this as 1-k to keep the output (y) within range.

Function 4
  Matrix
  1

Ensemble 4 (k,y) Hidden layer neurons for multiplication
  Vector 2
  Matrix
  1
  0

  Vector 3
  Matrix
  0
  1

End
## Table 2b, Linear Transformations

<table>
<thead>
<tr>
<th>Type</th>
<th>Relationship</th>
<th>Description</th>
</tr>
</thead>
</table>
| Vector   | \( \vec{Y}(t) = \mathcal{M}_{yx} \cdot \vec{X}(t) + \vec{D}(t) \) | Linear Coordinate Transformations  
Rotation \( \mathcal{M}_{yx} \) plus translation \( \vec{D}(t) \)  
Affine transformations \( \mathcal{M}_{yx} \)  
Cortical and motor output systems |
| Scalar Field | \( J(\vec{r}, t) = \int \vec{F}(\vec{r} - \vec{r'})I(\vec{r'}, t)d\vec{r'} \) | Linear Spatial Filters  
Lowpass or bandpass filters  
Wavelet transformation - Simple Cells |
| PDF      | \( \rho(\vec{Y}, t) = \int \rho(\vec{Y} | \vec{X})\rho(\vec{X}, t)d\vec{X} \) | Weighted conditional probability  
Generalized look up table (LUT)  
Marr-Albus theory of the cerebellum |
| Mixed    | \( \rho(D; \vec{r}, t) = \rho(D | I_L(\vec{r}, t), I_R(\vec{r}, t)) \) | Probability of Depth \( D \) at \( \vec{r} \) given left \( I_L(\vec{r}, t) \) and right \( I_R(\vec{r}, t) \) images. |
### Table 3a, Non-Linear Transformations

<table>
<thead>
<tr>
<th>Type</th>
<th>Relationship</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Scalar</strong></td>
<td>( Y(t) = \mathcal{P}[X(t)] = \sum_n \alpha_n X^n(t) )</td>
<td>Polynomial expansion&lt;br&gt;Static nonlinear feed-forward circuits</td>
</tr>
<tr>
<td><strong>Vector</strong></td>
<td>( \mathbf{Y}(t) = \mathcal{M}_{xy}(t) \mathbf{X}(t) )</td>
<td>Dynamic modulation of ( \mathcal{M}_{xy}(t) )&lt;br&gt;Control speed or frequency&lt;br&gt;Change between walk, trot and gallop&lt;br&gt;Modulate integrating time constants</td>
</tr>
<tr>
<td><strong>Scalar Field</strong></td>
<td>( J(\mathbf{r}, t) = \int \mathbf{F}(\mathbf{r} - \mathbf{r}' - \mathbf{s}(t)) J(\mathbf{r}', t) d\mathbf{r}' )</td>
<td>Translated Spatial Filters&lt;br&gt;Selective visual attention&lt;br&gt;Dynamic routing circuits</td>
</tr>
</tbody>
</table>
| **PDF**       | \( \rho(\mathbf{Y}, t) = \int \rho(\mathbf{Y}|\mathbf{X}, t) \rho(\mathbf{X}, t) \rho(\mathbf{X}, t) d\mathbf{X} \)  
\( = \int \rho(\mathbf{Y}|\mathbf{X}, \mathbf{Z}) \rho(\mathbf{X}, t) \rho(\mathbf{Z}, t) d\mathbf{X} d\mathbf{Z} \) | Time dependent conditional probability<br>Contextual Inference<br>Kawato's model of the cerebellum |
### Table 3b, Nonlinear Feedforward Neuronal Circuits

<table>
<thead>
<tr>
<th>Neuronal Coupling Weights</th>
<th>Explanation of Step</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scalar Polynomial Transform</td>
<td>Encode $Y(t)$</td>
</tr>
<tr>
<td>$b_m(t) = \mathcal{F}_m[Y(t)]$</td>
<td>Find weights $w_i^p$ by minimizing $\int[\mathcal{P}[X] - \sum_i w_i^p a(X)]^2 dX$</td>
</tr>
<tr>
<td>$\mathcal{F}_m[\sum_i w_i^p a_i(t)]$</td>
<td></td>
</tr>
</tbody>
</table>

#### Dynamic Vector Transformations

<table>
<thead>
<tr>
<th>Neuronal Coupling Weights</th>
<th>Explanation of Step</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b_m(t) = \mathcal{F}_m[\tilde{Y} \cdot \tilde{Y}(t)]$</td>
<td>Encode $\tilde{Y}(t)$</td>
</tr>
<tr>
<td>$\mathcal{F}_m[\tilde{Y}<em>m \cdot \mathcal{M}</em>{YX}(t) \cdot \tilde{X}(t)]$</td>
<td>$\tilde{Y}(t) = \mathcal{M}_{YX}(t) \cdot \tilde{X}(t)$</td>
</tr>
<tr>
<td>$\mathcal{F}_m[\tilde{Y}<em>m \cdot \sum_n \mathcal{M}</em>{YX}[n] \cdot \sum_i \tilde{X}_i a_i(t) c_n(t)]$</td>
<td>Decode $\tilde{X}(t)$ and $\mathcal{M}<em>{YX}(t)$, where $\mathcal{M}</em>{YX}(t) = \sum_n \mathcal{M}_{YX}[n] c_n(t)$</td>
</tr>
<tr>
<td>$\mathcal{F}<em>m[\sum</em>{n_k} w_{mn} a_i(t) c_n(t)]$</td>
<td></td>
</tr>
</tbody>
</table>

#### Dynamic Routing - Selective Attention

<table>
<thead>
<tr>
<th>Neuronal Coupling Weights</th>
<th>Explanation of Step</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b_m(t) = \mathcal{F}_m[\int \hat{\phi}_m(\bar{r}) J(\bar{r}, t) d\bar{r}]$</td>
<td>Encode $J(\bar{r}, t)$</td>
</tr>
<tr>
<td>$\mathcal{F}_m[\int \hat{\phi}_m(\bar{r}) \int F(\bar{r} - \bar{r} - \bar{s}) J(\bar{r}, t) d\bar{r} d\bar{s} \rho(\bar{s}, t)] d\bar{s}$</td>
<td>Shift Filter Center by $\bar{s}(t)$</td>
</tr>
<tr>
<td>$\mathcal{F}<em>m[\sum_i w</em>{mn} a_i(t) c_n(t)]$,</td>
<td>$\rho(\bar{s}, t) = \sum_n c_n(t) \phi_n(\bar{s}) \approx \delta(\bar{s} - \bar{s}(t))$</td>
</tr>
</tbody>
</table>

#### Dynamic Conditional PDF - Contextual Inference

<table>
<thead>
<tr>
<th>Neuronal Coupling Weights</th>
<th>Explanation of Step</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b_m(t) = \mathcal{F}_m[\int \hat{\phi}_m(\bar{Y}) \rho(\bar{Y}, t) d\bar{Y}]$</td>
<td>Encode $\rho(\bar{Y}, t)$</td>
</tr>
<tr>
<td>$\mathcal{F}_m[\int \hat{\phi}_m(\bar{Y}) \int \rho(\bar{Y}</td>
<td>\bar{X}, \bar{Z}) \rho(\bar{X}, t) \rho(\bar{Z}, t) d\bar{X} d\bar{Y} d\bar{Z}]$</td>
</tr>
<tr>
<td>$\mathcal{F}<em>m[\sum_i w</em>{mn} a_i(t) c_n(t)]$,</td>
<td></td>
</tr>
<tr>
<td>$w_{mn} = \int \int \int \hat{\phi}_m(\bar{Y}) \rho(\bar{Y}</td>
<td>\bar{X}, \bar{Z}) \phi_i(\bar{X}) \phi_n(\bar{Z}) d\bar{X} d\bar{Y} d\bar{Z}$</td>
</tr>
</tbody>
</table>
Table 2c, Feedforward Neuronal Circuits

<table>
<thead>
<tr>
<th>Neuronal Coupling Weights</th>
<th>Explanation of Step</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Vector Transformations</strong></td>
<td></td>
</tr>
<tr>
<td>$b_m(t) = \mathcal{F}_m[\hat{\mathbf{Y}} \cdot \hat{\mathbf{Y}}(t)]$</td>
<td>Encode $\hat{\mathbf{Y}}(t)$</td>
</tr>
<tr>
<td>$\mathcal{F}_m[\hat{\mathbf{Y}}<em>m \cdot \mathbf{M}</em>{yx} \cdot \hat{\mathbf{X}}(t)]$</td>
<td>$\hat{\mathbf{Y}}(t) = \mathbf{M}_{yx} \cdot \hat{\mathbf{X}}(t)$</td>
</tr>
<tr>
<td>$\mathcal{F}_m[\hat{\mathbf{Y}}<em>m \cdot \mathbf{M}</em>{yx} \cdot \sum_l \hat{\mathbf{X}}_l a_l(t)]$</td>
<td>Decode $\hat{\mathbf{X}}(t)$</td>
</tr>
<tr>
<td>$\mathcal{F}<em>m[\sum_l w</em>{ml} a_l(t)]$</td>
<td>$w_{ml} = \hat{\mathbf{Y}}<em>m \cdot \mathbf{M}</em>{yx} \cdot \hat{\mathbf{X}}_l$</td>
</tr>
<tr>
<td><strong>Filtered Scalar Field</strong></td>
<td></td>
</tr>
<tr>
<td>$b_m(t) = \mathcal{F}_m[\int \hat{\phi}_m(\vec{r}) J(\vec{r}, t) d\vec{r}]$</td>
<td>Encode $J(\vec{r}, t)$</td>
</tr>
<tr>
<td>$\mathcal{F}_m[\int \hat{\phi}_m(\vec{r}) \int \mathbf{F}(\vec{r} - \vec{r}') I(\vec{r}', t) d\vec{r}' d\vec{r}]$</td>
<td>Filter $I(\vec{r}, t)$</td>
</tr>
<tr>
<td>$\mathcal{F}_m[\int \hat{\phi}_m(\vec{r}) \int \mathbf{F}(\vec{r} - \vec{r}') \sum_l a_l(t) \phi_l(\vec{r}') d\vec{r}' d\vec{r}]$</td>
<td>Decode $I(\vec{r}, t)$</td>
</tr>
<tr>
<td>$\mathcal{F}<em>m[\sum_l w</em>{ml} a_l(t)]$</td>
<td>$w_{ml} = \int \hat{\phi}_m(\vec{r}) \mathbf{F}(\vec{r} - \vec{r}') \phi_l(\vec{r}') d\vec{r}' d\vec{r}$</td>
</tr>
<tr>
<td><strong>Weighted Conditional PDF</strong></td>
<td></td>
</tr>
<tr>
<td>$b_m(t) = \mathcal{F}_m[\int \hat{\phi}_m(\vec{Y}) \rho(\vec{Y}, t) d\vec{Y}]$</td>
<td>Encode $\rho(\vec{Y}, t)$</td>
</tr>
<tr>
<td>$\mathcal{F}_m[\int \hat{\phi}_m(\vec{Y}) \int \rho(\vec{Y}</td>
<td>\vec{X}) \rho(\vec{X}, t) d\vec{X} d\vec{Y}]$</td>
</tr>
<tr>
<td>$\mathcal{F}_m[\int \hat{\phi}_m(\vec{Y}) \int \rho(\vec{Y}</td>
<td>\vec{X}) \sum_l a_l(t) \phi_l(\vec{X}) d\vec{X} d\vec{Y}]$</td>
</tr>
<tr>
<td>$\mathcal{F}<em>m[\sum_l w</em>{ml} a_l(t)]$</td>
<td>$w_{ml} = \int \int \hat{\phi}_m(\vec{Y}) \rho(\vec{Y}</td>
</tr>
</tbody>
</table>