

Probing the structure-function relationship with neural networks constructed by solving a system of linear equations

Camilo J. Mininni^{1,2*}

B. Silvano Zanutto^{1,2}

1. Instituto de Biología y Medicina Experimental – Consejo Nacional de Investigaciones Científicas y Técnicas, Buenos Aires, Argentina.
2. Instituto de Ingeniería Biomédica – Universidad de Buenos Aires, Buenos Aires, Argentina.

*Corresponding author

mininni@dna.uba.ar (CJM)

Abstract:

Neural network models are an invaluable tool to understand brain function, since they allow to connect the cellular and circuit levels with behaviour. Neural networks usually comprise a huge number of parameters, which must be chosen carefully such that networks reproduce anatomical, behavioural and neurophysiological data. These parameters are usually fitted with off-the-shelf optimization algorithms that iteratively change network parameters and simulate the network to evaluate the changes and improve fitting. Here we propose to invert the fitting process by proceeding from the network dynamics towards network parameters. Firing state transitions are chosen according to the transition graph followed by an agent when solving a given behavioural task. Then, a system of linear equations is constructed from the network firing states and membrane potentials, in such a way that system consistency is guaranteed. This allows to uncouple the activity features of the model, like its neurons firing rate and correlation, from the connectivity features and from the task-solving algorithm implemented by the network, allowing to fit these three levels separately. We employed the method to probe the structure-function relationship in a stimuli sequence memory task, finding solution networks where commonly employed optimization algorithms failed. The constructed networks showed reciprocity and correlated firing patterns that recapitulated experimental observations. We argue that the proposed method is a complementary and needed alternative to the way neural networks are constructed to model brain function.

Introduction:

Understanding brain function requires construction of physiological models that explain experimental data, which encompass behavioural outcome, anatomical features, neurons biophysics and coding properties, among others^{1,2}. Many kinds of physiological models have been proposed along history, each one with their own merits. Among them, neural network models are well poised to connect all levels of analysis, from the behavioural to the molecular level, being a natural choice as neurons are the functional units of the brain. Yet, constructing neural networks that are suitable models is not an easy task. Neural networks can be hand-designed, setting network parameters following experimental data, or randomly chosen when experimental data is not available or as a mean of attaining more general conclusions. However, this approach may fall short given the complexity of the nervous systems. To tackle this issue, theorists have employed optimization methods to define the network parameters in such a way that a loss function is minimized. The loss function must encompass relevant aspects of the model, like its performance in one or several tasks, structural constraints such as the Dale's principle, or a connectivity with a certain degree of sparseness³. Optimization methods are widely used in artificial intelligence (AI), and the ongoing deep learning revolution has prompted an explosion of fitting algorithms, and the eagerness of taking advantage of them to build models of brain function^{4,5}. However, AI needs are different from the theoretical neuroscience needs. Artificial intelligence deals with constructing systems capable of solving difficult tasks, employing very general optimization algorithms parameter fitting⁶. On the other hand, models in neuroscience are expected to explain how animals behave in simple tasks, yet with biologically plausible neural networks. Simple tasks are desired because behavioural outcome is easier to interpret, and mechanistic explanations easier to envisage. Thus, in AI the difficulty strives in the task, while in theoretical neuroscience it strives in the restrictions in network design that are imposed by biology. Therefore, methods for parameter fitting in theoretical neuroscience can take leverage from this point – the simplicity of the task – to solve problems that could be too hard to solve with generic optimization methods.

One approximation that has been overlooked consists in finding the synaptic weights of a network as the solution of a system of equations. For many commonly employed neural network models, neurons perform a weighted sum of their inputs, followed by a non-linear transformation. For these models, if neurons firing and their added postsynaptic potential are known, the synaptic weights can be readily found by solving a linear system of equations in which the neurons firing constitute the coefficient matrix and the added postsynaptic potentials are the dependent variables. Thus, the problem of finding the network parameters is replaced by the problem of

finding sequences of valid network states that are consistent with solving the task. Although this problem might seem as hard as the former, we show in this work that viable network dynamics can easily be found by taking into account the transition graph associated with solving the task. By doing so, we were able to construct networks with millions of parameters extremely fast, without inefficient searches in parameter space. Moreover, optimization algorithms may have biases for a subset of all possible solutions⁷. These biases depend on the algorithms employed, the hyperparameters and the regularizations, and the relation between biases and its causes might be difficult to understand or control⁸. In contrast, our method allows to construct networks by sampling from a desired distribution of network dynamics, while further structural constraints on solutions can be easily imposed. Since the method proceeds from the network firing states to the network parameters, we call it the *Firing to Parameter* (FTP) method.

In this work we test the FTP method in a sequence memory task, and compare the method performance against an off-the-shelf optimization algorithm. Then, we show how to construct networks with certain activity and structural constraints, and analyse the relationship between structure and function.

Results:

Neural networks that follow a predefined transition graph:

We will consider an agent that interacts with its environment. At each time step t the agent is at one of M possible states m . Conversely, the environment adopts one of L possible states e . Agent and environment transitions can be expressed as:

$$[e_{t+1}, m_{t+1}] = [E(e_t, m_t), A(E(e_t, m_t), m_t)] \quad (1)$$

where E and A are the transition functions that take the agent and environment states and give the agent and environment states in the subsequent time step. The agent state m may codify several sub-states related to a biological agent, such as the behavioural response, reward signals, etc. Equation (1) thus describes a state machine which can model animal behaviour and neurophysiology. In particular, a behavioural task in which the agent must interact with the environment in a certain way to obtain reward can be codified in the E and A functions. Hence, any agent that solves a given behavioural task must follow the transition graph associated with solving that task. This includes agents controlled by recurrent neural networks, which are the main focus in this paper. We will work with networks of binary (McCulloch-Pitts) neurons composed of N_z recurrently connected integration neurons. Information about the environment is carried by a set of N_y sensory neurons (Fig. 1a). The temporal evolution of the network is dictated by:

$$\begin{aligned} u &= yw_y + zw_z \\ z &= H(u - \theta) \end{aligned}$$

where w_y and w_r are synaptic weights matrices of sensory and integration neurons, vector y contains firing states of sensory neurons, θ is a vector of neuron thresholds, and H stands for the Heaviside function. Vector u contains the neurons activation states, akin of membrane potentials, and vector z contains the neurons firing states. We will call z_m the vector of firing states of all integration neurons associated with population state m , and y_e the vector of firing states of sensory neurons associated with environment state e . Since we want our network to follow the state transitions depicted in eq. (1), the next equation must hold:

$$z_{A(e,m)} = H(c_{e,m}w - \theta) \quad (2)$$

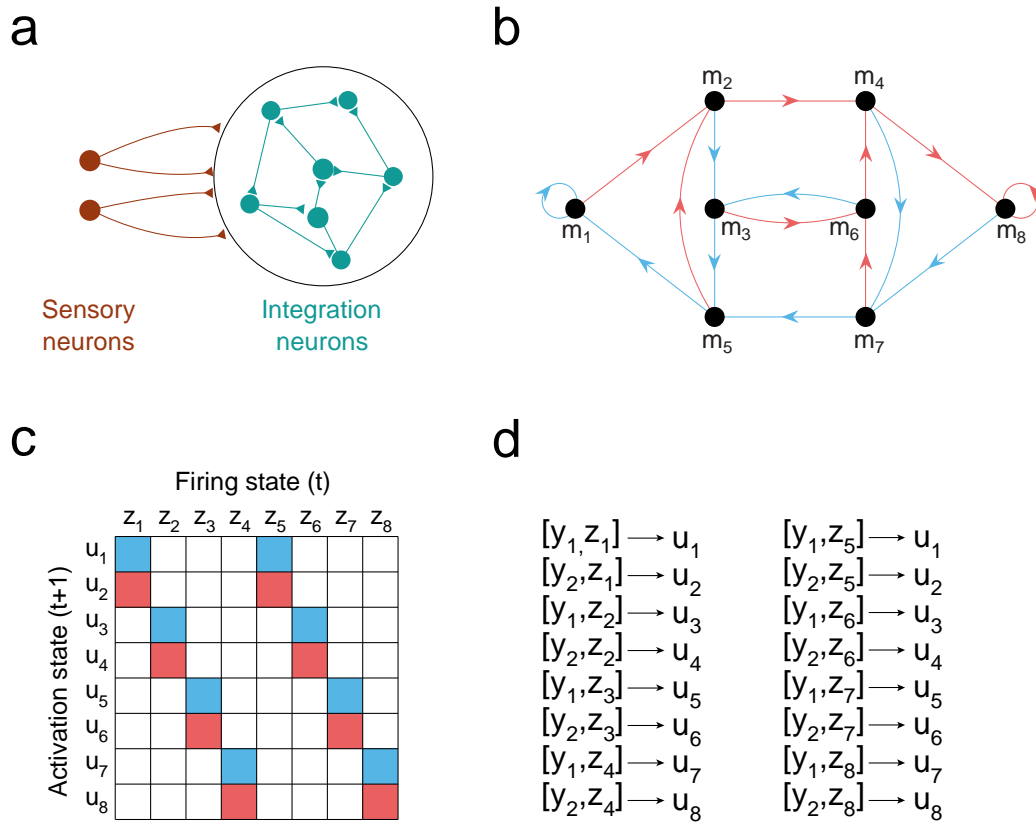


Figure 1. Constructing recurrent networks that follow a predefined transition graph. **a**, networks are composed of binary McCulloch & Pitts neurons. Sensory neurons codify external stimuli and project to the integration neurons through synaptic weights w_y . Integration neurons are recurrently connected through synaptic weights w_r . In the s-task, the population of integration neurons must codify in its firing state the sequence of the last τ stimuli presented. **b**, transition graph showing transitions between network states during execution of the s-task, for $\tau=3$. Each node in the graph is a network state, and arrows depict transitions between nodes after stimuli presentation. Each possible sequence of 3 stimuli in length is codified by exactly one network state. Nodes are numbered such that transitions can be represented in a simple transition matrix. **c**, transition matrix associated with the transition graph in panel (b). It shows the activation states u that are reached when recurrent neurons are in a population firing state z , and s_1 (blue) or s_2 (red) are presented. **d**, same transitions depicted in panels (b) and (c), but explicitly showing vectors u_i and vectors c_i , which are the concatenation of one y and one z . The index i is such that z_i and u_i are the firing state and activity vectors corresponding to agent state m_i .

where $c_{e,m} = [y_e, z_m]$ is the concatenation of sensory and integration neurons firing states, and $w = [w_y, w_z]$ is the concatenation of the synaptic weight matrices. Equation (2) says that transitions in network states as ruled by the weight matrix w must be consistent with transitions in the transition graph that solves the target task.

Linear system construction:

We are interested in constructing a recurrent network of neurons by solving a system of linear equations built from the set of neurons firing states. To this end we first define a coefficient matrix C , whose rows result from the concatenation of one y_e vector and one z_m vector, for all combinations of environment state e and agent state m :

$$C = \begin{pmatrix} y_{e_1} & z_{m_1} \\ \vdots & \vdots \\ y_{e_L} & z_{m_1} \\ \vdots & \vdots \\ y_{e_1} & z_{m_M} \\ \vdots & \vdots \\ y_{e_L} & z_{m_M} \end{pmatrix}$$

Then we define a matrix U , such that its i th row vector u_i is such that $H(u_i - \theta)$ gives the z associated with the state m that should follow from the e state and m state of row i in C , according to the desired transition graph. Thus, matrix C and U condense all the transitions required to solve the target task. It follows that:

$$Cw = U \quad (3)$$

The matrix w can be found by computing:

$$w = C^+U \quad (4)$$

where C^+ stands for the pseudoinverse of C . Thus, the connectivity matrix w can be obtained by solving the linear system with coefficient matrix C and dependent variable matrix U . Since we are using the pseudoinverse to solve the system, the solution is the one with minimum Frobenius norm⁹.

We want to sample from the set of matrixes w that accomplish the task constrain, i.e. networks constructed with the sole constrain of solving the target task. A naïve approximation to this problem would be to pick the u_i vectors at random, threshold them to obtain the associated z_i vectors, and construct matrices C and U by following the desired transition graph. However, by doing so it is very likely that we end up by having an inconsistent system of equations, meaning

that there is no network of neurons that can follow those state transitions. This is because matrix C is not full rank, but its rows are linearly dependent. If we consider the case of two stimuli s_1 and s_2 , codified by vectors y_1 and y_2 , then each vector $[y_1, z_m]$ can be expressed as a linear combination of $[y_2, z_m]$ and vectors $[y_1, z_p]$ and $[y_2, z_p]$, where vector z_p can be any vector taken from the set of all firing states the network can adopt:

$$[y_2, z_m] = [y_1, z_m] - [y_1, z_p] + [y_2, z_p] \quad (5)$$

Thus, $\text{rank}(C) = M + 1$. Following the Rouché-Capelli theorem¹⁰, eq. (3) has a solution if and only if $\text{rank}(C) = \text{rank}([C, U])$, being $[C, U]$ the augmented matrix. Yet, if we choose vectors u randomly, when adjoined to matrix C the linear dependencies expressed in eq. (5) will be broken, and the resulting augmented matrix will have rank above $M + 1$. However, consistency can be enforced if initial randomly generated vectors u are linearly combined following linear dependencies in C , such that the same linear dependencies in C are conserved in the augmented matrix.

The s-task:

In the following we will consider a sequence memory task: the environment consists of two stimuli s_1 and s_2 , which are sequentially presented at each time step, chosen randomly with equal probability. To obtain reward at time step t the agent has to recall the stimulus presented at time step $t - \Delta t$. Successful behaviour thus requires to have a memory of stimuli sequences of length $\tau = \Delta t + 1$, starting from $t - \Delta t$. The constant τ defines the memory requirements of the task. Figure 1b shows an agent's states and the transitions gated by the stimuli when solving the s-task for $\tau = 3$.

To solve the task the agent needs at least $M = 2^\tau$ states, meaning that complexity grows exponentially. This would suggest that the task is a complex one. However, it can be seen that the transition matrix has a stereotyped form if nodes are numbered properly (Fig. 1c). The transition graph in Fig. 1b shows the state transitions any agent that solves the s-task should follow. With the transition matrix structure at hand we can construct matrices C and U . We will define a neural network with two sensory neurons such that $y_{s_1} = (1, 0)$ and $y_{s_2} = (0, 1)$. If we order the transitions as in Fig. 1b-d, we have that, if eq. (3) has a solution, rows in the augmented matrix should satisfy:

$$[y_2, z_{T+1}, u_{T+1}] = [y_1, z_T, u_T] - [y_1, z_P, u_P] + [y_2, z_{P+1}, u_{P+1}] \quad (6)$$

where T are indexes over the rows of C and U , which are odd numbers between 1 and M . The row index P is an odd number between 1 and M , different from all T . Note that $z_T = z_{T+1}$ and $z_P = z_{P+1}$, but $u_T \neq u_{T+1}$ and $u_P \neq u_{P+1}$. Equation (6) shows us how row vectors in matrix U should be linearly combined such that eq. (3) has a solution. We have that:

$$u_{T+1} = u_T - u_P + u_{P+1} \quad (7)$$

This means that the number of linear combinations in U is $R = 2^r / 2 - 1$, and $\text{rank}(U) = 2^r / 2 + 1$. Note that rewriting eq. (7) we have:

$$u_{s_2,i} - u_{s_1,i} = \Delta_i \quad (8)$$

where $u_{s_1,i}$ and $u_{s_2,i}$ are the activation that neuron i adopts after presentation of s_1 and s_2 , respectively. In words, eq. (8) tells that the difference in effects provoked by the stimuli is a constant for each neuron, regardless of which network state or transition we are dealing with. This fact is not surprising, since synaptic weights are held fixed, so each stimulus has the same effect at any time, which is specific for each neuron. Thus, making the system of equations in (3) consistent is equivalent to guarantee that activation values are chosen so that the effect of each stimulus is consistent.

We proceeded by generating a vector of thresholds θ , with $\theta_i \in \{\frac{1}{2}, \frac{3}{2}, \frac{5}{2}\}$. Then, we constructed base matrix U_{base} with $M_{base} = 2^r / 2 + 1$ row vectors such that:

$$U_{base}(m,i) = \theta_i + r(m,i) + \frac{1}{2}$$

where $r(m,i)$ is an integer uniformly sampled from the $[-5,5]$ interval. We added the term $\frac{1}{2}$ to avoid fitting errors when numerically solving the system, otherwise the activation values could be equal to the threshold values, which would result in erroneous firing states because of numeric precision issues. This initial randomly generated matrix U_{base} has full rank. We computed the vector Δ of Δ_i elements as the difference between the first two rows of U_{base} . Next, we applied eq. (8) to generate the remaining R rows as linear combinations of the third to the last row of U_{base} , obtaining 2^r row vectors which constitute the matrix U^* . Each row vector u in matrix U^* has the neurons activations for one of the 2^r network state. Applying eq. (8) creates a

dependency between $u_{s_1,i}$ and $u_{s_2,i}$. Hence, for each linear combination we chose at random which activation value (the one associated with s_1 or s_2) will be defined in terms of the other. This is to ensure that u value distributions are equal between stimuli. We constructed matrix Z by applying threshold θ to U^* , and then we followed the ordering depicted in Fig. 1b-d to construct matrix U from U^* , and matrix C from Z and vectors y_1, y_2 . Finally, we employed eq. (4) to obtain the synaptic weight matrix w . Since matrix w is the minimum Frobenius norm solution to eq. (4) and defines a network that solves the s-task, we call it a $T+F$ network.

We are assuming that two conditions are met after thresholding: 1) the resulting vectors c are all different, and 2) they are linearly independent. If after thresholding any vector is repeated, this would result in lower performance in the task, since not all sequences of length τ will be encoded. On the other hand, if linear independency fails after thresholding, then matrix C will have more linear combinations than the contemplated in eq. (5), meaning that combining the rows of U following eq. (8) will not be enough, and some linear dependencies in C will be lost in the augmented matrix, making the system inconsistent. In our implementation of the algorithm, if any of these two conditions were not verified, then the algorithm was restarted from the beginning. This occurred sometimes, for $\tau < 5$. For higher τ , both conditions were always fulfilled in one attempt.

In the above explanation we assumed that $N_z = 2^\tau$, such that there is one neuron per sequence of length τ . It was possible to fit networks with lower number of neurons, but undesired linear dependencies in C after thresholding, or a number of network states below 2^τ occurred with higher probability, especially for $\tau > 3$.

We employed the FTP algorithm to construct networks of $N_z = 2^\tau$ neurons that solve the s-task (Fig. 2a,b). The resulting synaptic weight distribution had zero mean and resembled a normal distribution, at least for the w_r values (Fig. 2c). In fact, the synaptic weight distributions became progressively closer to a normal distribution as more neurons were used in network construction (Fig. 2d). We also noted that the absolute weight value decreased, especially for w_r values (Fig. 2e), which can be explained by thinking that more neurons imply more parameters and hence more degrees of freedom to reach a lower Frobenius norm. This observation will become important later when imposing structural constraints to the network.

Efficiency of the FTP algorithm:

We assessed the performance of the algorithm by measuring the time expended in finding solutions for $\tau = 1$ to $\tau = 12$, and comparing these times with the times required for a genetic algorithm (GA) to find solutions for the same τ values and number of neurons. The FTP

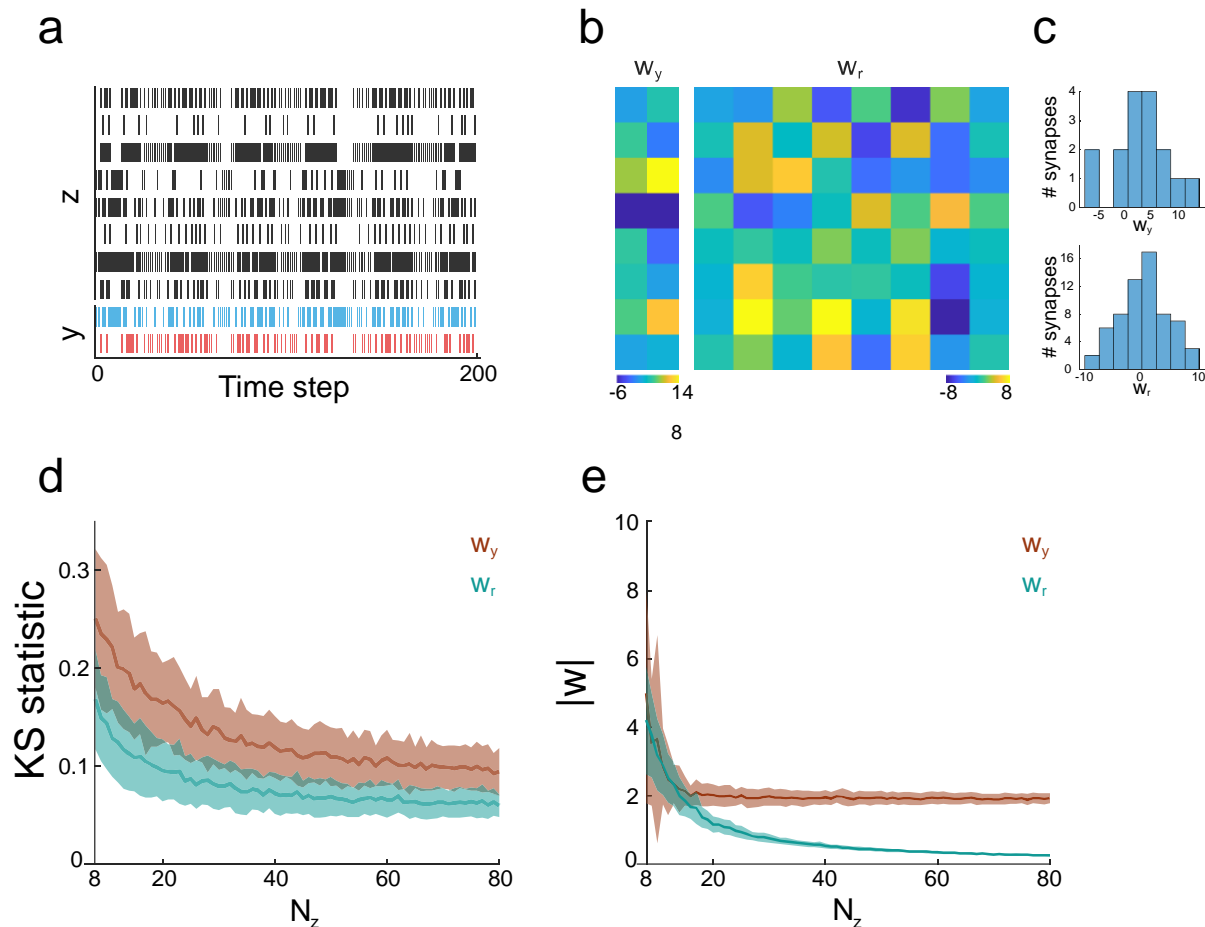


Figure 2. Solving the s-task with the FTP algorithm. **a**, raster plot showing the neurons firing states in a network constructed to solve the s-task for $\tau = 3$. The network is composed of eight integration neurons and 2 sensory neurons. Each possible sequence of 3 stimuli has a unique network firing state that codifies it. Therefore, the network has 8 possible firing states. **b**, sensory and integration synaptic weights of the network. For this example ($f_r = 1$) the range of weights is greater for sensory than for integration synapses. **c**, distribution of synaptic weights for sensory (upper panel) and integration (lower panel) synaptic weights, for the same network as in (a, b). Distributions are zero centred. **d**, Kolmogorov-Smirnov statistic between the distribution of synaptic weights and a normal distribution of the same mean and variance. As the number of integration neurons increases, the distribution of synaptic weights gets closer to a normal. Integration neurons weights are closer to a normal than sensory neurons weights. Mean \pm SD are shown for $n = 100$ networks that solves the s-task, with $\tau = 3$. **e**, absolute synaptic weight values as a function of the number of integration neurons in the network. Absolute values are higher and of larger variability when the neuron count is close to the number of coded stimuli sequences. As the number of integration neurons increases the absolute mean value and dispersion decreases. Sensory neurons weights quickly reach a minimum, while integration neurons weights decrease in the entire range of integration neurons. Mean \pm SD are shown for $n = 100$ networks that solves the s-task, with $\tau = 3$.

240

241

242

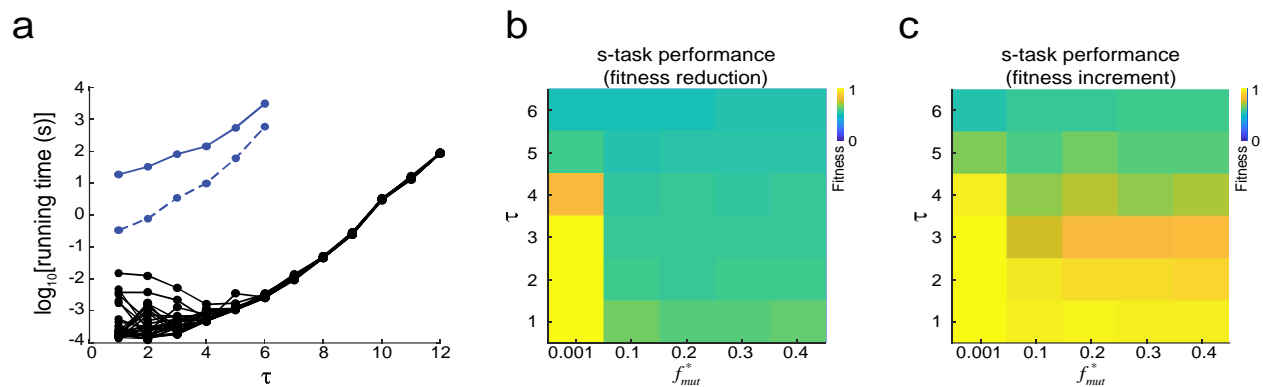


Figure 3. Efficiency of the FTP algorithm. **a**, efficiency of FTP and GA, measured as the time expended in finding a network solution for s-tasks of different τ . The time expended by the FTP algorithm is orders of magnitude lower than the time expended by the GA. As τ increases, the time expended in network simulation (dashed blue line) tends to match the total time expended by the GA (solid blue line). The FTP running time showed some variability for $\tau < 6$, in the range of tens of milliseconds. The absence of points in the GA curves for $\tau > 6$ means that this algorithm could not find a solution within the limit of 1 hour of running time. The GA was run 1 time for each τ , while the FTP was run 30 times for each τ . **b**, fitness of solution networks found for different τ after their synaptic weights were mutated for 20 generations with different mutation ratios f_{mut}^* , and selected to reduce their fitness. Mutations reduced fitness to chance level for all f_{mut}^* values except for the lowest. For $\tau \geq 5$ even the lowest f_{mut}^* had detrimental effects. **c**, fitness measured after networks in (b) where subjected to 20 generations of mutation and selection to increase their fitness. Fitness could be restored when τ and f_{mut}^* values were low. If τ or f_{mut}^* were higher, restoration was only partial or did not occurred.

243 outperformed the genetic algorithm by several orders of magnitude (Fig. 3a), and for $\tau > 6$ the
 244 GA could not find a network with performance above 0.9. The result is not surprising, since the
 245 time complexity of solving a linear system of equations is $O(n^3)$ ¹⁰, with n being the total number
 246 of variables. As τ was increased, most of the GA running time was expended in the simulation
 247 of the networks, while the fraction of time expended in evaluating network performance with the
 248 classifier became smaller.

249 Given that performance of optimization algorithms is sensitive to several hyperparameters,
 250 including initial conditions and mutation factor, we asked whether a GA that starts near a solution
 251 network would stay around the solution or would it drift away. To that end we employed the FTP
 252 to find a solution network, and set it as the individual from which the first population was built.
 253 Then the GA was used to reduce fitness over 20 generations, such that the population became
 254 20 generations apart from the solution (Fig. 3b). Next, the GA was run for another 20 generations
 255 in the increasing fitness direction (Fig. 3c). In each generation, random mutations were applied,
 256 with a mutation factor $f_{mut} = f_{mut}^* \sigma(w)$ (see Methods). We run this experiment for several values
 257 of τ and f_{mut}^* . It can be seen that up to $\tau = 4$ performance dropped to chance level during the
 258 fitness reduction phase, followed by a total or partial recovery during the fitness increment phase.
 259 However, performance did not recover for higher values of τ . This suggests that, as τ increases,

the basin of attraction around the initial solution gets narrower, leading the GA to drift away from the solution. Although these results cannot rule out that other optimization algorithms have better performance, they do highlight how small the solution space is, and the huge gap between the FTP and another, more generic fitting algorithm.

Imposing activity constrains through U matrix initialization:

We want to construct neural network models that not only solve relevant tasks but do so under desired firing constrains, as measured in real brains. Some of these constrains are low firing rates (FR)^{11,12}, or low correlation coefficient (CC)¹³. In regular optimization algorithms, these constrains can be imposed to solution networks by introducing regularization terms in the loss function³. On the other hand, in the FTP algorithm the activity states of the network are the result of linearly combining the rows of an initial matrix U_{base} . Hence, we can apply firing constrains by appropriately choosing this initial matrix. For example, to attain networks that solve the s-task with low/high firing it suffices to choose an initial matrix U_{base} such that after thresholding the resulting matrix C has few/many ones. Following this procedure, we constructed networks with average FR within a wide range of target FR (Fig 4a, blue line). Shuffling the afferent synaptic weights of each neuron produces only small changes to the average FRs (Fig. 4a, red line). This suggests that it is the distribution of afferent synaptic weights the critical structural statistic that defines the networks average FR, and not its precise connectivity. Solutions are harder to find for extreme FR values, because thresholding gives C matrices with repeated rows, which translate in not enough network states to codify all stimuli sequences.

On the other hand, we can construct networks with desired signal correlation, by multiplying Δ by a factor f_{cc} , which results in stimuli inducing different firing rates (Fig. 4b). For networks shown in Fig. 4 ($\tau = 4, f_r = 3$) correlations could be modulated in a range between 0.25 and 0.5. Although scaling of Δ is expected to induce signal correlation, it can be seen that it is inducing noise correlation as well, as a by-product (Fig. 4c). Correlations of networks solving the s-task are significantly higher than correlations of their synaptic weights-shuffled counterparts (Fig 4b, blue vs. red), which shows that pairwise correlations depend on the whole weight matrix and not only on the distribution of the afferent weights, as is the case with FR. It also suggests that the set of networks that solve the s-task necessarily have correlation above a minimum. On the other hand, correlations also seem not to exceed a certain value: higher correlations would imply a reduced number of network states, incompatible with the number of sequences required to codify.

Hand-based manipulation of U_{base} allows to generate solution networks in a wide range of FR and CC (Fig. 4d, blue dots). An even better control of firing and correlation can be achieved by

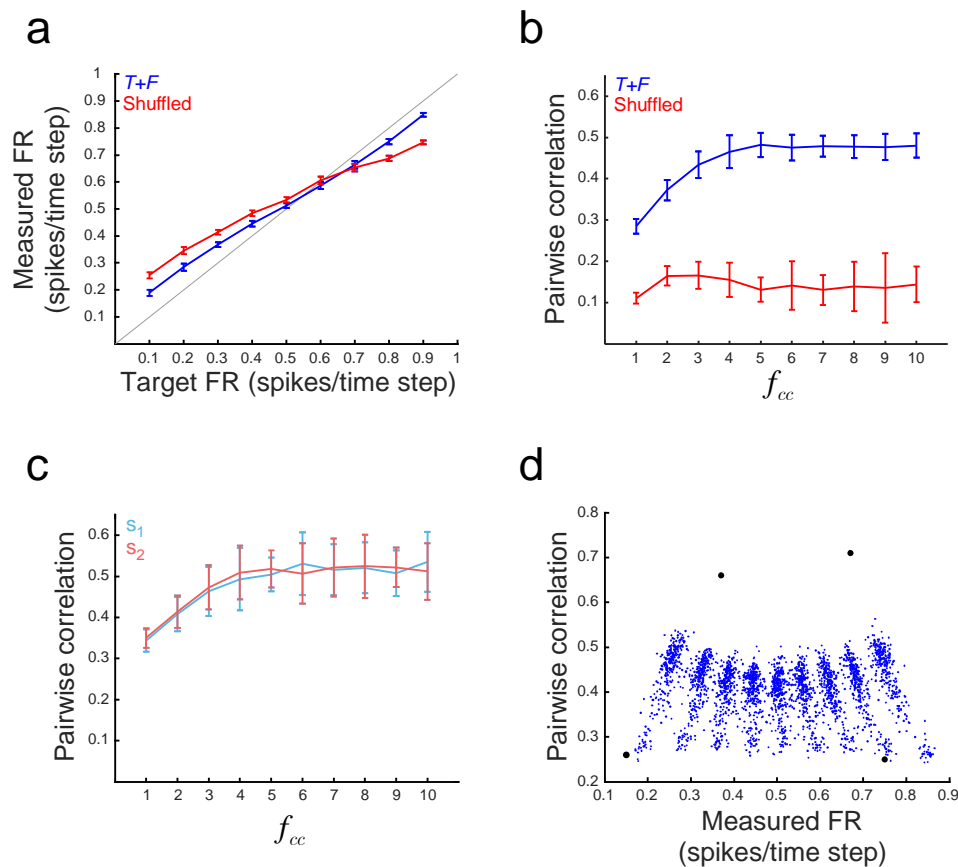


Figure 4. Using FTP to construct networks subjected to task and activity constraints. **a**, real FR measured in networks constructed to solve the s-task, as a function of the target FR. The FR of networks constructed with the FTP algorithm are close to the target FR (blue line). There is a tendency to obtain lower firing rates for target FR values above 0.5 spikes/ time step, and higher firing rates for target FR values below 0.5 spikes/time step (grey line is the identity function). The same networks with their afferent synaptic weights shuffled (red line) show a similar relationship between target FR and measured FR, albeit with a lower slope. A total of 30 networks were generated for each target FR. Mean \pm SD are shown, $n = 30$. **b**, correlation between pairs of integration neurons as a function of the scaling factor f_{cc} . Pairwise correlation, computed over all time steps, increases with f_{cc} until it saturates at $CC = 0.48$ for $f_{cc} \geq 5$ (blue line). Networks with their afferent synaptic weights shuffled (red line) show low correlation, invariant to f_{cc} . A total of 30 networks were constructed for each f_{cc} value, with target FR set to 0.1 spikes/time step. Mean \pm SD are shown, $n = 30$. **c**, pairwise correlation computed separately for s_1 and s_2 (noise correlation). The correlation coefficient increases with f_{cc} , similarly for both stimuli, and closely following correlation values in (b). Mean \pm SD are shown, $n = 30$. **d**, Measured FR as a function of pairwise correlation. Each blue dot shows the FR and CC of one network constructed to solve the s-task with FR and correlation constraints imposed by U_{base} initialization. Values for 2700 networks are shown. Points form stripes pointing towards FR = 0.5 spikes/time step, each stripe corresponding to networks with the same target FR. As correlation increases, the measured FR tends to 0.5 spikes/time step. Black dots show FR and correlation of 4 networks for which FR and correlation constraints were imposed by evolution of a population of U_{base} matrices.

294 fitting U_{base} by means of a GA in which the fitness of U_{base} is a function of the FR and CC
 295 computed over the population firing states of the network generated from that U_{base} . Fitting U_{base}
 296 allows for more extreme values of FR and CC (Fig 4d, black dots), while keeping computational

cost low by computing FR and CC over the set of population firing vectors c instead of computing the actual network activity by simulating the network. Altogether, both methods (U_{base} manipulation, or its evolution with a GA) easily allow to generate networks that perfectly solve the task, while imposing desired activity constrains at the same time.

Applying structural constrains with projected gradient descend in isofunction weight space:

Networks generated so far share one structural constrain: their synaptic weights matrix is the one that minimizes the Frobenius norm. Other relevant structural constrains, such as the lack of self-connections, Dale's principle, or sparse connectivity are not satisfied. Since these structural constrains are key experimentally observed features (Lefort et al. 2009; Seeman et al. 2018; Strata and Harvey 1999) but see^{17,18}, we were interested in imposing such constrains onto the w obtained by the algorithm. To do this we followed a projected gradient descent (PGD) approach¹⁹, taking advantage of the fact that the loss function \mathcal{L} , which encloses the structural constrains, is a linear function with respect to the synaptic weights, and that the matrix w can be changed without changing the stimulus-response mapping (see Methods). To exemplify the procedure we constructed a network that solves the s-task for $\tau = 4$, with $f_r = 3$ (Fig. 5a), and then we employed PGD to transform its matrix w to remove self-connections, enforce Dale's principle with a 4:1 Ex:In ratio, and set a sparsity $sp = 40\%$ (defined as the percentage of weights equal to zero). The PGD reduced the loss function \mathcal{L} in a steady fashion, reaching a negligible error, provided that the network had enough neurons (Fig. 5b). It is remarkable how such different synaptic weight matrices, as the ones depicted in Fig. 5a,c,d, gave rise to exactly the same stimulus-response mapping.

We noted that structural constrains could not be imposed to networks with low number of neurons, i.e. $N = N_y + N_z$ between M and $3M$. This is not surprising, since it is expected that imposing more constrains requires more parameters. To evaluate the efficiency of the PDG in relation with the number of neurons, we imposed the above structural constrains for networks solving the s-task with $\tau = 3$ to $\tau = 6$, and N_z between 32 and 256 neurons. Since matrix U and vector θ were randomly chosen, it is expected that some of them result in matrices w for which the structural constrains are impossible to apply. Consequently, we measured PDF efficiency by computing $\# attempts$, the number of networks that were required to generate until obtaining the first successfully constrained network. It can be seen that $\# attempts$ decreased as the number of neurons increased (Fig. 5e). Concordantly, the computing time required to obtain a fitted network decreased as the number of neurons increased, because fewer fitting attempts

331

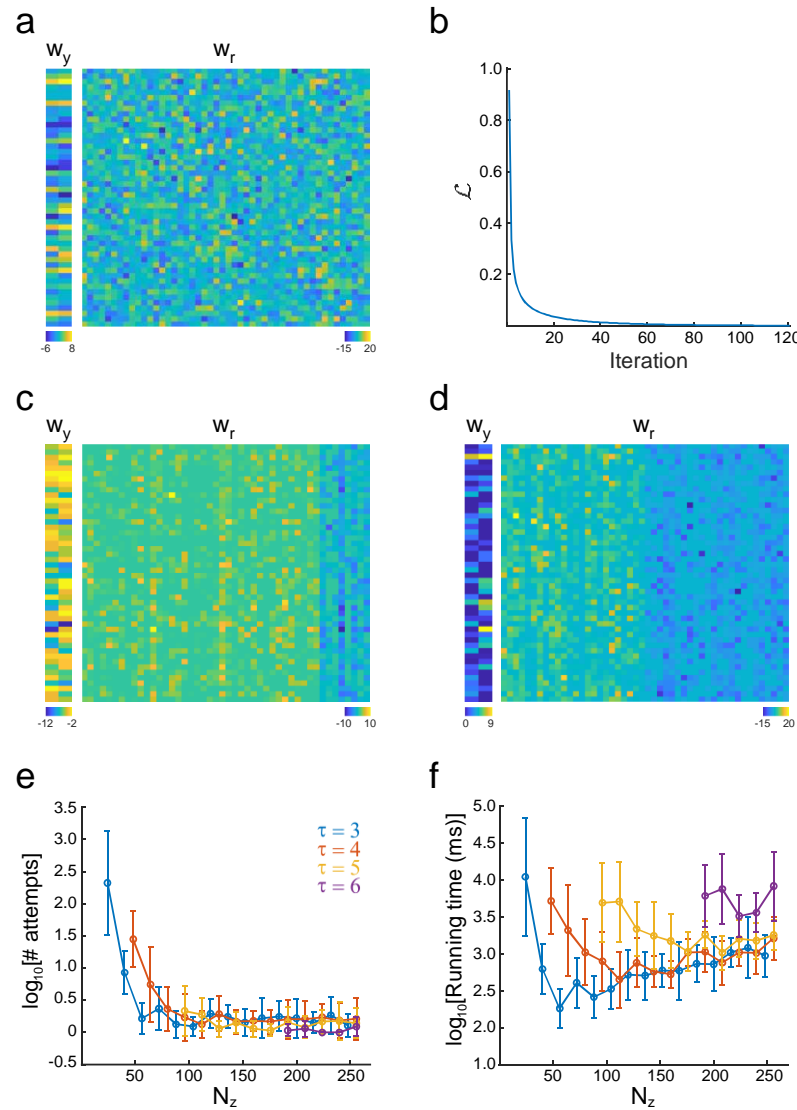


Figure 5. Applying structural constraints to networks. **a**, synaptic weight matrices w_y and w_r of the network obtained through FTP before structural constraints were imposed. The network was constructed to solve the s-task for $\tau = 4$ and $f_r = 3$, with a target FR of 0.1 spikes/time step. **b**, Loss function \mathcal{L} as a function of the number of iterations of the PGD algorithm. The loss function falls below the criterium $e_1 = 10^{-3}$ at iteration 121. **c**, **d**, synaptic weight matrices w_y and w_r for a network with the same stimulus-response mapping but after applying structural constraints: (c) no self-connection, Dale's principle, with 40 excitatory and 10 inhibitory neurons, and sparsity $sp = 40\%$; (d) no self-connections, Dale's principle, with 26 excitatory and 24 inhibitory neurons, and sparsity $sp = 23\%$. **e**, average number of attempts to obtain one network with successful structural fitting, as a function of the number of integration neurons, and for different τ . The number of attempts is high when the neuron number is low, but it decreases fast as the neuron number increases. From 60 neurons onwards, less than five attempts are needed, on average, to obtain one network with the desired structural constraints. Mean \pm SD are shown. **f**, total running time to obtain one network with successful structural fitting, as a function of the number of integration neurons, and for different τ (color code as in (g)). Running time decreases and then increases for $\tau = 3$ and $\tau = 4$. The case of $\tau = 6$ is the one with more neurons and equations to solve, and present some of the highest running times, even when the number of neurons is high. Nevertheless, all average running times are below the tens of seconds.

were required (Fig. 5f). The fitting time was somewhat higher for networks with the highest neuron count, but always within the order of tens of seconds, even for $N_z = 256$.

Linking structure, function and activity:

Neural network structure determines its activity, which in turn translates into function. To understand the relationships between these three network features we constructed networks with different average firing rate and functionality and analysed their structure, more precisely, their connectivity. One key aspect of connectivity is reciprocity, which has been observed experimentally²⁰ and its implications studied theoretically²¹. Here we chose the correlation between weights of incoming and outgoing synapses as the measure of reciprocity²² (see Methods).

We have already shown how the FTP algorithm can be employed to generate networks with predefined activity features, namely, with desired firing rate and correlation. To compare networks with different *functionality* we constructed networks which had the same number of neurons and network states but for which the graph of transitions between network states was generated at random (Fig. 6a,b). In this manner we can construct networks whose dynamics have complexity similar to that of networks that solve the s-task, but which lack their function, i.e. to codify sequences of stimuli of length τ . We screened networks with memory ranging from $\tau = 2$ to $\tau = 7$, and FR from 0.1 spikes/time step to 0.9 spikes/time steps, and found that the reciprocity varied with τ , FR and neuron number. In particular, we observed that, when $f_r = 1$, reciprocity was positive and of lower mean for networks that were the minimum Frobenius norm solution to the s-task ($T+F$ networks, Fig. 6c), in comparison with networks that were the minimum Frobenius norm solution to a random transition graph (F networks, Fig. 6d). However, for bigger f_r the relationship was inverted, and $T+F$ networks showed positive reciprocity (Fig. 6e) while F networks showed negative reciprocity (Fig. 6f).

To further describe these relationships, we selected networks constructed for $\tau = 7$ and $f_r = 1$ (Fig. 7a-c) and $f_r = 4$ (Fig. 7d-f). Signal and noise correlation varied with FR following an inverted U-shape relationship, with a maximum next to 0.5 spikes/time step (Fig. 7a,d). Note that, up to a FR = 0.5 spikes/time step, CC increased with FR, as has been observed experimentally^{23,24}. Interestingly, the way CC changed with FR was similar for both $T+F$ and F networks, with the distinction that F networks had overall higher correlations than $T+F$ networks. We computed the CC after shuffling the inter-spike interval of each neuron and found that it remained practically invariant with respect to FR. These CC values were also much smaller than the CC values of the non-shuffled firings ($CC_{\text{shuffled}} = 0.0224 \pm 1.10^{-4}$, mean \pm SD, for $n = 90$ $T+S$ networks pooled over

366

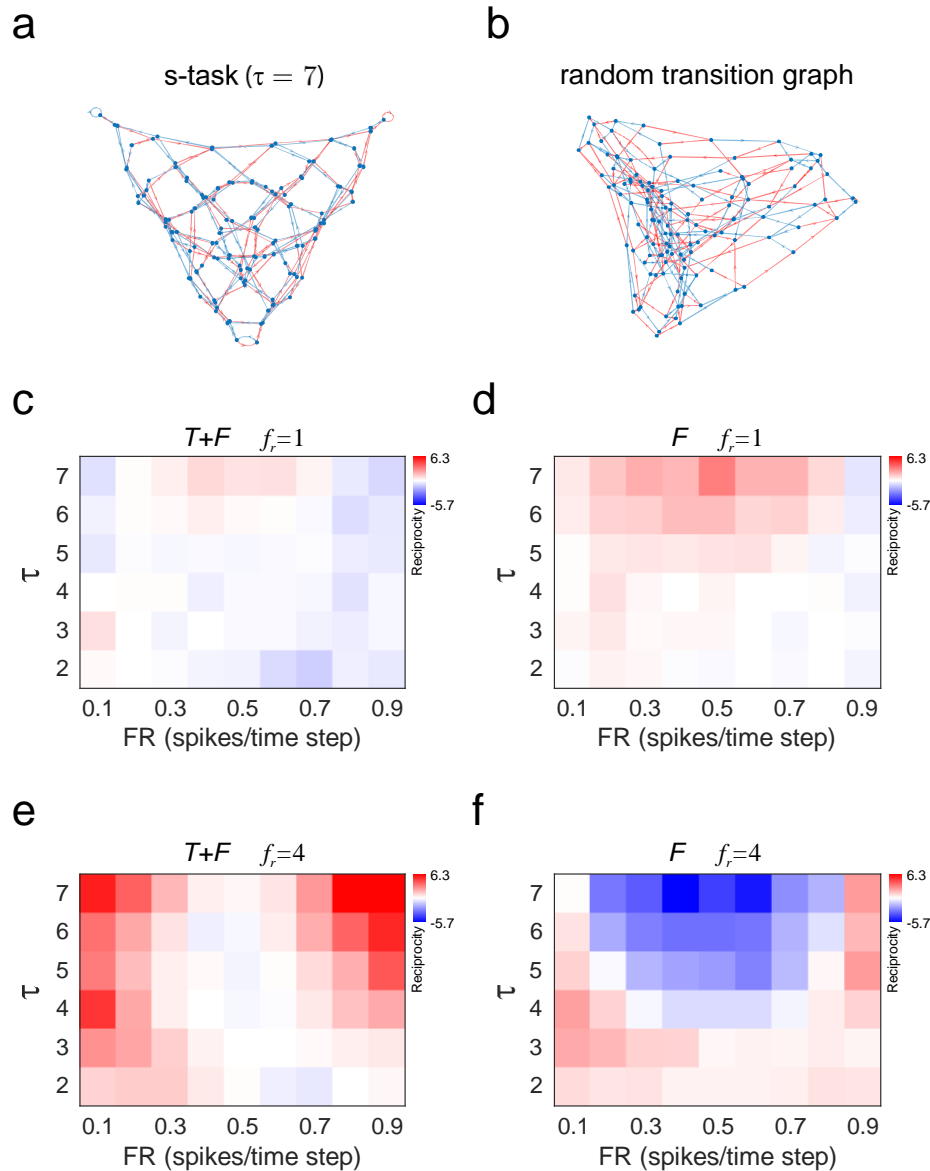


Figure 6. Reciprocity as a function of τ , FR and the type of transition graph. **a**, transition graph for solving the s-task with $\tau = 7$. Blue and red lines represent transitions gated by s_1 and s_2 , respectively. **b**, random transition graph. Nodes (network states) may receive different number of incoming connections. There are 24 nodes that are gated by both stimuli. **c**, reciprocity for $T+F$ networks, with $f_r=1$, as a function of τ and target FR. Reciprocity changes from slightly negative to slightly positive as τ increases. For $\tau = 7$, reciprocity is maximized around target FR = 0.5 spikes/time step, and decreases for lower and higher values of target FR. **d**, F networks with $f_r=1$ shows increasing positive reciprocity as τ increases, maximized at target FR = 0.5 spikes/time step. **e**, when the number of neurons is higher ($f_r=4$), $T+F$ networks show positive reciprocity that is minimal around target FR = 0.5, and increases towards higher and lower target FR, reaching the highest reciprocity values among all networks screened. **f**, reciprocity of F networks gets increasingly negative as τ increases, reaching the lowest reciprocity among all networks screened, around target FR = 0.5 spikes/time step. For all panels, 30 networks were constructed for each τ and target FR combination. Normalized means (mean/SD) are shown. Positive and negative reciprocity values were mapped separately to colours red and blue, respectively. Red tones go from 0 reciprocity (white) to maximal positive reciprocity (pure red). Blue tones go from 0 reciprocity (white) to maximal (in absolute value) negative reciprocity (pure blue). All random graphs were constructed with $f_{bc}=0.5$. Graphs were plotted with the Force-directed layout

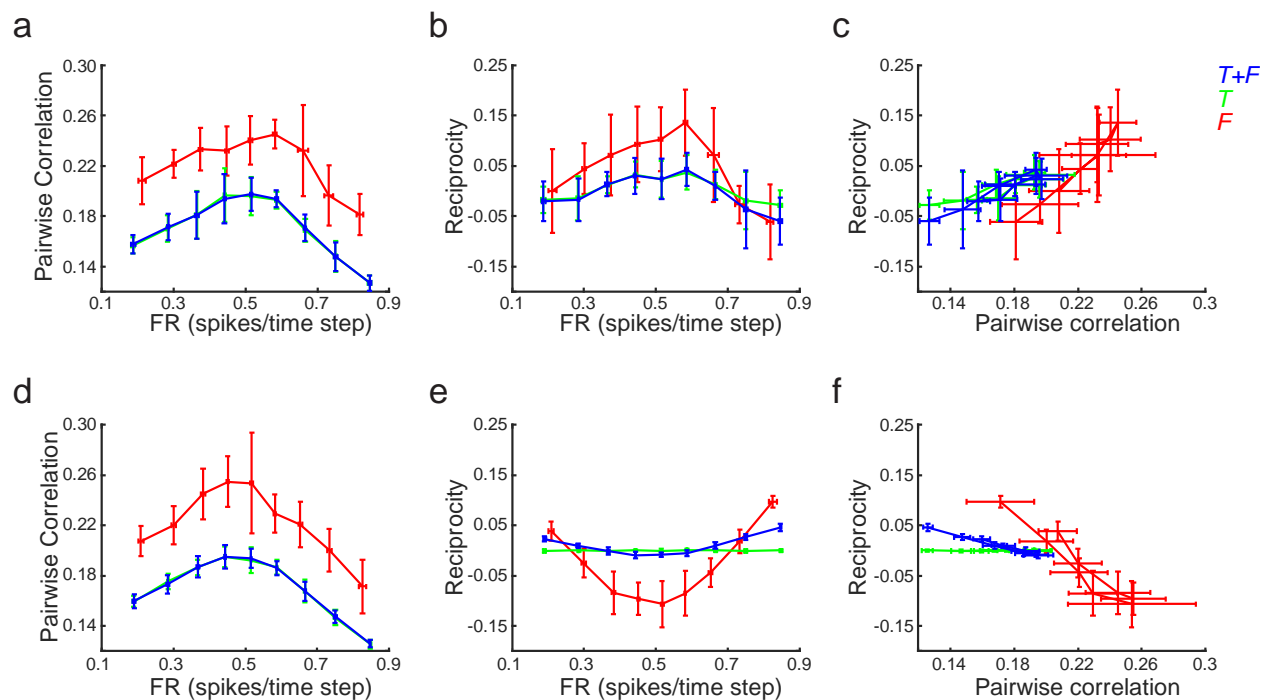


Figure 7. Correlation and reciprocity differentiate networks with sequence memory from random transition networks. **a-c**, networks constructed to solve the s-task with $\tau = 7$ and $f_r = 1$ ($T+F$ networks), their isofunction network (T network), and networks with the same number of neurons and network states that follow a random transition graph (F networks). (a), correlation increases as FR approaches 0.5 spikes/time step. F networks show a positive offset with respect to $T+F$ and T networks. (b), the dependency between reciprocity and FR is similar to the dependency between CC and FR. Higher reciprocity values are found in F networks. (c) reciprocity grows linearly with correlation, as expected from panels (a) and (b). **d-f**, idem a-c, but with $f_r = 4$. (d), the CC/FR relationship is similar to the one observed with lower neuron number (panel (a)). **e**, the reciprocity/FR relationship inverted as the neuron number was increased. Reciprocity is minimized as FR approaches 0.5 spikes/time steps, and increases towards lower or higher FR values. F networks show pronounced negative reciprocity. (f) reciprocity decreases linearly with correlation, as expected from panels (d) and (e). Mean \pm SD are shown; $n = 10$ networks were constructed for each target FR and network type. All random graphs were constructed with $f_{bc} = 0.25$.

all FR values). These results rule out the possibility that correlations were trivially increasing with FR because of the higher number of spikes. We also found an inverted U-shape between reciprocity and FR, and a linear relationship between reciprocity and CC. The parabolic relationship is accentuated in networks with more neurons, the curve being more pronounced and of lower dispersion. With $f_r = 1$ reciprocity tended to be maximized as FR approached 0.5 spikes/time step, with F networks showing higher (and positive) reciprocity (Fig. 7b). With $f_r = 4$, reciprocity tended to increase as FR departed from 0.5 spikes/time step towards lower and higher values, i.e., networks with lower correlation (Fig. 7e). Specially, F networks showed negative reciprocity for all firing rates, except for the more extreme cases (0.1 and 0.9 spikes/time step). Just as the reciprocity/FR relationship inverts with the number of neurons, so does the reciprocity/CC relationship. Networks with higher reciprocity have higher correlation when the

number of neurons is low (Fig. 7c). However, and somewhat counterintuitive, when the number of neurons is higher, more reciprocity implies lower correlation (Fig. 7f). Networks that solve the s-task but do not minimize the Frobenius norm (T networks) showed almost zero reciprocity. This implies that reciprocity is not a property of all networks that solve the s-task. On the contrary, most networks that solve the s-task do not show significant reciprocity, unless other structural constrain, such as Frobenius norm minimization, is imposed. However, Frobenius norm minimization alone only produces negative reciprocity (in random graphs). For positive reciprocity to occur in networks with high number of neurons, both high sequence memory and Frobenius norm minimization is required.

We asked whether the results depicted in Fig. 7 also occur in networks which lack self-connections and comply with Dale's principle. To that end we imposed these structural constraints to networks constructed with $\tau = 7$ and $f_r = 4$, and found a reciprocity/FR relationship that resembles the one observed in unconstrained networks, with F networks showing prominent negative reciprocity and $T+F$ networks showing increasing reciprocity as FR departs from 0.5 spikes/time step (Fig. 8a). Correlations increased as FR approached 0.5 spikes/time step, with F networks showing more correlation than $T+F$ networks (Fig. 8b,c). Correlation in $T+F$ and F networks were higher for pairs of inhibitory neurons than for pairs of excitatory neurons, as has been observed experimentally²⁵, while correlations between excitatory and inhibitory neurons laid in the middle. It is interesting to note that the classification of neurons as excitatory or inhibitory was not defined by design, but emerged during the enforcement of the structural constraints, when the network firing states were already chosen. This suggests that it was the (predefined) firing statistic of the neurons, specially the correlation among them, which ultimately defined which neuron could become excitatory and which inhibitory.

In the F networks studied so far, each network state can be reached from either one of the two stimuli, or from both stimuli. This is the case because the random transition graphs allow nodes with incoming edges from both stimuli. Network states which can be reached from both stimuli (bicoloured nodes, see Methods) codify stimuli in a relative manner, meaning that the identity of the stimulus presented at time step t can be decoded if the network state at time step t and at time step $t+1$ is known. On the other hand, network states which can be reached from exclusively one of the two stimuli (monocoloured nodes), codify stimuli in an absolute manner, since it is possible to know the identity of the presented stimuli at time step t by knowing the network state at time step $t+1$ alone. We asked whether the proportion of relative coding states and absolute coding states could explain the strong differences in correlation and reciprocity found between $T+F$ and F networks. To that end, we changed the fraction of nominal bicolored

nodes f_{bc} and computed reciprocity for F networks of fixed τ , FR and f_r (Fig. 9). We found that negative reciprocity values are caused by relative coding network states, since reciprocity is reduced as the fraction of these states is increased. When all network states are absolute coding states ($f_{bc} = 0$), reciprocity is the lowest, as observed in $T+F$ networks with the same FR and f_r . This suggests that reciprocity differentiates networks by how their network states codify stimuli, regardless of the capacity of the network for sequence coding.

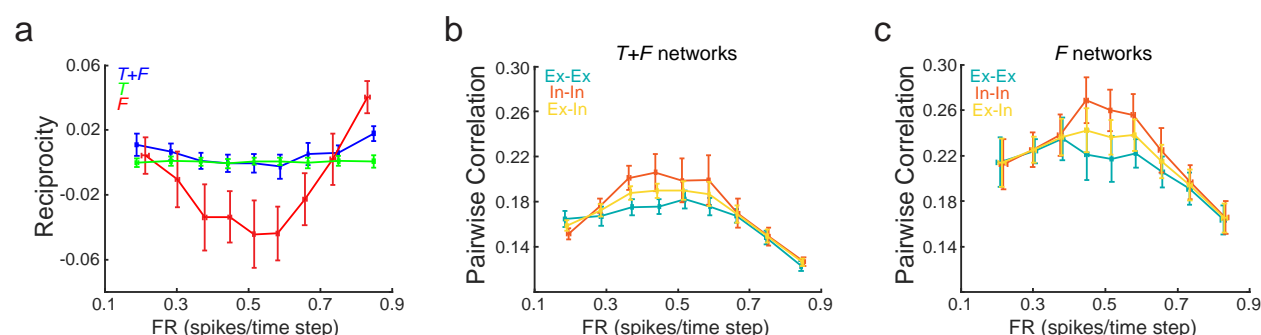


Figure 8. Reciprocity and correlation of structurally constrained networks. **a**, reciprocity as a function of FR for networks without self-connections and Dale's principle with 1:1 Ex:In ratio. Reciprocity shows a parabolic relationship with FR, decreasing as FR approaches 0.5 spikes/time step. F networks show strong negative reciprocity, while T networks reciprocity is close to zero. **b**, pairwise correlation for $T+F$ networks as a function of firing rate. Correlation was computed over pairs of excitatory neurons (Ex-Ex), pairs of inhibitory neurons (In-In), and pairs of one excitatory and one inhibitory neuron (Ex-In). Correlation has a maximum close to 0.5 spikes/time step. The In-In pairs show the highest correlations, followed by the Ex-In pairs. The Ex-Ex pairs show the lowest correlation. **c**, pairwise correlation for F networks as a function of firing rate. The CC/FR relationship is similar to the one observed for $T+F$ networks, although F networks correlation is displaced towards higher values. Mean \pm SD are shown; $n = 20$ networks were constructed for each target FR and network type. Firing rates of excitatory or inhibitory neurons are displayed for Ex-Ex and In-In curves, respectively. For Ex-In curves the average FR over all neurons is shown.

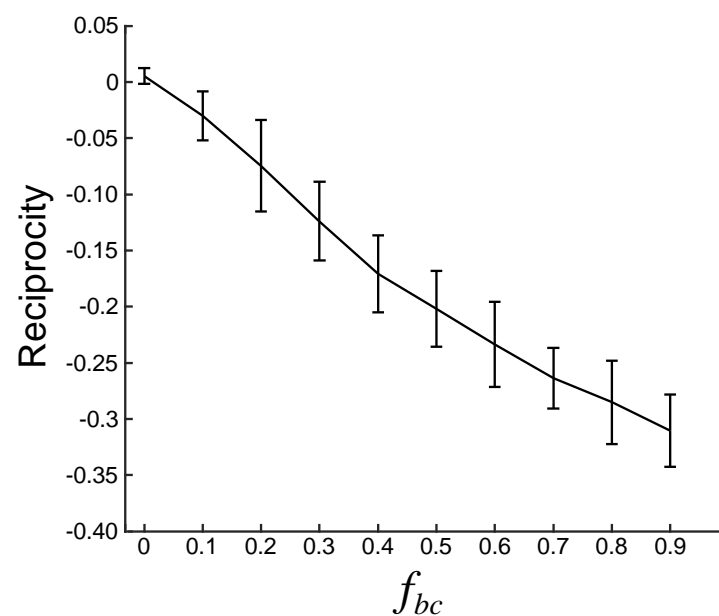


Figure 9. Relative coding network states cause negative reciprocity. Reciprocity as a function of f_{bc} , the fraction of nominal bicolored nodes, in networks that follow random transition graphs. Reciprocity decreases linearly with f_{bc} , approaching zero as f_{bc} approaches zero. Mean \pm SD are shown, $n = 30$ networks for each f_{bc} . Networks were constructed with target FR = 0.5 spikes/ time step, and with the same number of neurons and network firing states as $T+F$ networks constructed with $\tau = 7$ and $f_r = 4$.

Discussion:

We have presented a simple method to generate binary neural network models that accomplish a desired task. Binary networks are computationally inexpensive, and despite their simplicity many neurophysiological and neuroanatomical observations have been recapitulated by means of these networks^{21,26}. Our key contribution is to note that, for networks in which neurons inputs are linearly added, their synaptic weights can be found by solving a system of linear equations. In turn, this system can be constructed from the transition graph associated with the solution of the target task. System consistency is guaranteed if the dependent variables of the system (the neurons activations) are linearly combined following the linear dependences among the independent variables (the firing states). We have shown how the FTP method works with the simplest of networks. Yet, we think the same procedure can be implemented in networks built from more complex neuron models, like the firing rate model or the leaky integrate-and-fire model, provided that a system of linear equations can be constructed.

Current automated methods for constructing network models rely on off-the-shelf optimization algorithms typically employed in the artificial intelligence field, like stochastic gradient descent³, genetic algorithms²⁷ or evolutionary strategies²⁸. These optimization algorithms iteratively change network parameters in a direction that minimizes a loss function, and have proved to be very effective in finding networks that solve very complex tasks^{29,30}. However, they require a considerable amount of human design, and there are no guarantees that they can reach a solution. Moreover, each optimization iteration requires the evaluation of the network, which is time consuming, especially for a recurrent network performing in a multi-trial task. In contrast, the FTP algorithm reduces the problem of finding a suitable network to a series of linear combinations and the solution of a linear system, which can be solved in polynomial time. Most importantly, it is guaranteed that the resulting network will solve the task perfectly.

When employing traditional optimization algorithms to fit neural networks, a loss function is defined, taking into account all the required constraints, whether these are task related, activity related, or structural. Then, the loss function is minimized and hence all constraints are enforced at once. In this scenario the relationship between parameters and the loss function can be quite complex, and some conflict between constraints may emerge. Conversely, one key advantage of our method is that it allows to uncouple the dynamic and coding aspects of the network from the structural aspects, giving the opportunity of sampling them independently. Since the method proceeds from the firing states to the parameters, it allows to find networks with desired activity profiles, and to study the resulting connectivity. Further structural constraints can be enforced in a second stage, by projected gradient descent, or any other optimization algorithm. The fact that

projected gradient descent worked so well suggests that structural constraints are easy to implement once the connectivity required to solve the task is in place. This is probably because structural measures such as sparsity, clustering, etc are a simple function of the synaptic weights. On the contrary, the relationship between weights and network function or coding capabilities are more complex. In this way, our method gives more control over each constraint type, making the whole process simpler at the same time.

To show the applicability of the method we employed it to construct networks that solve a stimulus sequence memory task (s-task) in which the network has to codify in its network firing states the sequence of the last τ stimuli that were presented. This task is relevant in the broad sense of working memory function. Although working memory is traditionally associated with maintaining information about a single stimulus in the persistent activity of recurrently connected neurons^{31,32}, mounting evidence suggests that neuron populations code information in the form of highly heterogeneous firing sequences^{33,34}. Sustained activity can be a suitable strategy when there is one specific relevant stimulus to attend, whose identity has been already elucidated. However, more complex scenarios require keeping track of sequences of stimuli. An example of this case is the processing of language, in which the succession of utterances must be integrated over time, from phonemes, to words, to phrases, so that the meaning of speech depends on the whole sequence³⁵. We explored the case of two stimuli presented with equal probability, but the analysis could be extended to more realistic cases in which stimuli presentation probability is not uniform. It is expected that statistical regularities in the input sequences are going to be exploited by the network, resulting in more specialized connectivities. The relationship between sequence statistics and network structure should be further studied. For example, it would be of interest the case in which each stimulus lasts more than one time-step, and they are interleaved by neutral stimuli, which could act as distractors. Then, the relationship between feedback and feedforward connections could be studied, in relation with the duration of each stimulus presentation, and with that of the distractors.

The structure-function relationship is central to neuroscience^{36–38}. Connectivity at the macro meso and micro scale, neurons biophysics, plasticity mechanisms, among other structural traits, all act co-ordinately to give sophisticated adaptive behaviour. It is widely believed that structural properties of networks have evolved over time to proficiently perform function, many times in an optimal way^{39,40}. However, brain structure could also be the result of other constraints, different from those imposed by adaptive behaviour. For example, neural network modularity might have emerged as a good structural trait for solving tasks which have a modular or hierarchical aspect⁴¹. But it could also have emerged as a result of previously acquired structural traits such as

constrains in the length of dendrites and axons, which precludes the possibility of a much wider connectivity. Thus, determining how much of the structure observed in the brain comes from task-related constrains and how much comes from other structural traits is central to understanding the structure/function relationship. A theoretical approximation to this issue consists in constructing neural network models that solve different kind of tasks under a variety of structural constrains, and then study the pattern of connectivity that emerges and relate it to the experimentally observed connectivity in real brains. This approximation requires to sample as uniformly as possible from the set of networks that fit the task and structural constrains. However, optimization methods commonly used in network parameter fitting may give a restricted set of solutions, thus biasing any conclusion about the structure/function relationship. Another issue is that some connectivity traits could emerge only in networks of certain size, and fitted to several tasks. In this case, fitting large networks with complex cost functions could have a high computational cost. Consequently, generating a relatively large sample of networks suitable for statistical treatment of their connectivity would result unfeasible. In this aspect, the FTP algorithm is very well suited for answering structure/function questions, since exact solutions can be computed starting from an arbitrary set of population firing codes, as long as it defines a system of equations that have a solution.

The FTP approach allows to test hypothesis linking structure and function by constructing networks which follow transition graphs that instantiate some null hypothesis. Following this approach, we constructed networks which had the same number of network states and neurons required to solve the s-task, but whose state transitions were chosen at random. With this tool at hand we were able to show that a structural feature emerges as τ and redundancy increase, evidenced in the reciprocity of the network. The same procedure can be followed to build any other set of networks in accordance with some relevant null hypothesis. Such networks can be easily constructed with the FTP method, while they would be hard to construct with regular optimization algorithms.

Evidence for high reciprocity has been found experimentally, by measuring excitatory postsynaptic potentials of reciprocally connected neurons in vitro²⁰. It has also been the centre of theoretical analysis. For example, it has been shown that high reciprocity is recapitulated in networks of binary neurons that have maximum number of attractors²¹. Interestingly, the same work shows that reciprocity is lost when networks are optimized to remember sequences of uncorrelated network states. However, we showed that, when networks are built to codify sequences of stimuli, the network itself shows sequences of states that follow the sequences of stimuli up to an arbitrary τ .

Reciprocity was absent in networks taken at random from the set of all connectivities that give the same dynamics. This implies that the observed reciprocity is the result of following a particular transition graph with the additional constrain of weights minimizing the Frobenius norm, the latter being explained biologically as an upper bound on the size of the synapses. Thus, to explain one structural feature (reciprocity), a functional feature (solving the s-task) and another structural feature (Frobenius norm minimization) were required. It would be interesting to study to what extent other structural features encountered in biological neural networks, like modularity or sparsity of connections, can be explained as the answer to some computational demand of adaptive behaviour, or they are the result of another structural feature, or both factors interact, as is the case of the s-task.

In conclusion, we have provided a method that inverts the usual process of constructing neural network models. It allows to probe the dependency between the firing statistics, connectivity and function of a network in a way that is not matched by current optimization algorithms. Moreover, it is computationally inexpensive. Therefore, we consider the method to be a powerful alternative to the way neural networks are constructed to model brain function.

Methods:

Network simulation and synaptic weights statistics:

Networks were evaluated in the s-task during at least $N_{iter} = 10.2^\tau$ time steps, to gather enough samples of each network state. To assess the similarity between the synaptic weight distribution and a normal distribution we computed the Kolmogorov-Smirnov two samples statistic, between the set of synaptic weights and a set of normally distributed values of the same mean, variance and sample size than that of the synaptic weights.

Equation (4) gives the matrix w with lowest Frobenius norm¹⁰. Since we are considering networks with $N = 2^\tau + 2$ total neurons (including sensory and integration neurons), there are infinite solution matrices w for the same system of equations defined by C and U . These solutions lay in a subspace of \mathbb{R}^N , of dimension $N - \text{rank}(C)$. The set of all solutions can be obtained by finding a matrix Δw such that:

$$C\Delta w = 0 \quad (9)$$

$$\Delta w = \ker(C)\mathcal{M} \quad (10)$$

where $\ker(C)$ is an orthonormal basis of the null space of C of dimensions $N \times (N_z - \text{rank}(C))$, 0 is a matrix of zeros, and \mathcal{M} is a linear mapping of dimensions $(N_z - \text{rank}(C)) \times N_z$. These networks share the same stimulus-response mapping. We say they conform an isofunction space.

For several applications, networks with $N_z > 2^\tau$ were desired. Hence, we defined $N_z = 2^\tau f_r$, where f_r stands for 'redundancy factor', as the network has f_r -times more neurons than required to solve the s-task with that specific τ .

Computation of fitting times for FTP and a genetic algorithm:

We assessed the efficiency of the FTP algorithm by measuring the time expended in finding networks that solve an s-task with $\tau = 1$ to $\tau = 12$, and $f_r = 1$.

We also computed the time expended by a genetic algorithm (GA) to obtain networks that solve the s-task of $\tau = 1$ to $\tau = 10$. We employed a population of $N_{pob} = 200$ individuals, each one composed by one matrix w_y , one matrix w_r and one vector θ . Networks were evaluated in the s-task for $N_{iter} = 10.2^\tau$ time steps, and its fitness F was defined as 1 minus the classification loss of a support vector machine, trained to classify the stimulus presented at time step $t - \tau + 1$ based on the population firing state at time step t . The classification model was cross-validated with the holdout method, trained on 50% of the data and tested on the remaining 50%. We picked

$T = 0.1N_{pob}$ individuals with the highest fitness as parents. Then, we picked parents at random and built the next generation by mutating each synaptic weight with gaussian noise of zero mean and standard deviation $f_{mut} = 0.1\sigma(w)$. The factor $\sigma(w)$ is the standard deviation of the synaptic weights in w_y of networks generated with FTP if we are mutating w_y , and the standard deviation of synaptic weights in w_r if we are mutating w_r . We defined f_{mut} in this way to avoid f_{mut} values that are so big that a solution can not be reached, or so small that the solution will not be reached in a reasonable amount of time. One of the individuals of each generation was an unmutated copy of the best individual of the previous generation (elitism). Threshold vectors θ were not mutated. The GA was run until the average fitness surpassed $F_{target} = 0.9$, or after 60 minutes of search. Once the stopping criterion was met, the elite individual was evaluated during 100.2^τ time steps to obtain the final fitness.

We also tested how stable was a solution obtained with FTP under evolution with a GA. We employed FTP to construct networks of given τ and $f_r = 1$. From this network a population of $N_{pob} = 100$ individuals was constructed by mutating the network with a mutation rate $f_{mut} = f_{mut}^* \sigma(w)$, where $\sigma(w)$ is the standard deviation of the synaptic weights obtained with FTP. As before, w_y and w_r has their own f_{mut} , according to their corresponding standard deviation. Then, a GA was employed to reduce fitness during 20 generations, and then to increment it for another 20 generations, employing the same f_{mut}^* . We followed the procedure for τ ranging from $\tau = 1$ to $\tau = 6$, and $f_{mut}^* = 0.001$ to $f_{mut}^* = 0.4$ in steps of 0.1. We performed 10 repetitions for each τ and f_{mut}^* combination.

Imposing activity constrains:

To construct networks with desired FR we generated U_{base} as described in the Results Section, but adjusted the sign of $r(m, i)$ such that, after thresholding, matrix C had a fraction of ones equals the target FR. To induce signal correlation, we scaled vector Δ by a factor f_{cc} . This manipulation makes each neuron to have very different firing rates for s_1 and s_2 , which increases the signal correlation. By following this procedure, we constructed networks in Fig. 4. The target FR values were taken from the range between 0.1 to 0.9 spikes/time step, in steps of 0.1 spikes/time step. The f_{cc} values were taken from the range between 1 and 10 in unitary steps. A total of 30 networks were constructed for each combination of FR and f_{cc} values within those ranges. For each network the average FR was computed over the FR of all neurons in the network. Similarly, the average correlation coefficient (CC) was computed from the Spearman correlation coefficient computed for all neuron pairs.

We also employed a GA to evolve a population of U_{base} matrices to fit their mean FR and correlation. We employed a population of $N_{pob} = 200$ individuals, each one composed of one matrix U_{base} and one vector θ . For each individual we constructed U and C matrices, and computed an approximate value of FR and correlation, under the assumption that each network firing state occurs with equal probability. The fitness F of an individual was computed as

$$F = 1 - \frac{|FR - FR_{target}| + |CC - CC_{target}|}{2}$$

where FR and CC are the firing rate and correlation values of the networks output, and FR_{target} and CC_{target} are the firing rate and correlation values we want the networks to have.

If an individual produced an inconsistent system, or a system with not enough network states, its fitness was set to zero. We chose $T = 0.1N_{pob}$ and mutated U_{base} by adding gaussian noise to each matrix element, of zero mean and standard deviation $\sigma = f_{mut} = 0.1$. Threshold vectors θ were not mutated. Elitism was employed. The GA was run until the average fitness surpassed $F_{target} = 0.95$. Firing rates and correlations shown in Fig. 4d were computed by running the network constructed from the elite U_{base} during 30.2^f time steps.

Imposing structural constrains:

Solving equation (4) gives networks with minimum Frobenius norm. These networks do not suffice basic structural features observed experimentally, such as the lack of self-connections or Dale's principle. To impose such structural constrains we constructed a matrix Δw_d such that $w_d = w + \Delta w_d$. Matrix w_d is a matrix which fulfils the desired structural constrains. Most probably Δw_d will not be within the null space of C , and thus w_d will not be a solution to the system defined by C and U . Hence, we defined a matrix:

$$\Delta w = \ker(C) \mathcal{M}_{sc} \quad (11)$$

where $\mathcal{M}_{sc} = \ker(C)^+ \Delta w_d$, and $\ker(C)^+$ is the Moore-Penrose pseudoinverse of $\ker(C)$. Matrix \mathcal{M}_{sc} is a linear mapping that incorporates the desired structural constrains, making Δw the change in matrix w within the null space of C that is closest to Δw_d , in the least squares sense.

We imposed three structural constrains: no self-connections, Dale's principle, and a certain degree of sparsity. Thus:

$$\Delta w_d = \Delta w_{self} + \Delta w_{Dale} + \Delta w_{sp}$$

The matrix Δw_{self} (which deletes self-connections in the integration neurons) has values:

$$\Delta w_{self}(j, i) = \begin{cases} -w(j, i) & j = i + 2 \\ 0 & j \neq i + 2 \end{cases}$$

Matrix Δw_{dale} was defined as:

$$\Delta w_{Dale}(i, j) = \begin{cases} 0 & (T_j = Ex \wedge w(i, j) > 0) \vee (T_j = Inh \wedge w(i, j) < 0) \\ -w_{c_1}(i, j) & (T_j = Ex \wedge w(i, j) < 0) \vee (T_j = Inh \wedge w(i, j) > 0) \end{cases}$$

where $w_{c_1} = w + \Delta w_{self}$ and $T_j \in \{Ex, Inh\}$ indicates if neuron j was chosen to be excitatory (Ex) or inhibitory (Inh). Matrix Δw_{Dale} sets to zero the synaptic weights that violate Dale's principle. Neuron j was chosen to be excitatory if $\eta_j = \sum_i w(i, j) > 0$. Otherwise, it was chosen to be inhibitory. If more excitatory/inhibitory neurons were required, neurons with negative/positive η closest to 0 were set as excitatory/inhibitory as needed.

Matrix Δw_{sp} to enforce sparsity was defined as:

$$\Delta w_{sp}(i, j) = \begin{cases} -w_{c_2}(i, j) & w_{c_2}(i, j) < \alpha(sp) \\ 0 & otherwise \end{cases}$$

where $w_{c_2} = w_{c_1} + \Delta w_{Dale}$. The value $\alpha(sp)$ is the sp percentile of the absolute values in w_{c_2} . In this manner Δw_{sp} will set to zero the lowest weights such that a sparsity sp is enforced.

The loss function $\mathcal{L}(k)$ at iteration k was defined as the average of the absolute $\Delta w_d(i, j)$ values:

$$\mathcal{L}(k) = \langle |\Delta w_d(i, j)| \rangle_{i,j}$$

where $\langle \rangle_{i,j}$ stands for the average across indexes (i, j) . The structural constraints were imposed through an iterative process, in which at each iteration the neurons were classified as excitatory or inhibitory according to their η , a matrix Δw was computed using eq. (11), and a

new w was obtained. The process was stopped when the loss fell below a desired value e_1 , in which case the fitting process was considered successful. The process was also stopped if $(L(k) - L(k-1)) / L(k) < e_2$. When this latter condition was met, the fitting process was considered unsuccessful, since the error was not decreasing fast enough and would probably converge to an unacceptable value above zero. We used $e_1 = 10^{-3}$ and $e_2 = 10^{-4}$. If the process was successful, values that violated any of the constraints were clipped to zero. These values are expected to be small enough since the error was small. We computed:

$$e_{clip} = \langle |U_{sc}(i, j) - U(i, j)| \rangle_{i,j}$$

where $U_{sc} = Cw_{sc}$ with w_{sc} being the resulting synaptic weight matrix after the constraining process, to verify that the deviation from the original U was negligible. If the process was unsuccessful, or the clipping error $e_{clip} > 10^{-3}$, then the original w was considered not to be suitable for the structural fitting.

We measured the efficiency of the process by computing the number of networks generated (# *attempts*) and the running time t expended until reaching the first successfully constrained network. We varied τ from $\tau = 3$ to $\tau = 6$. For each τ we varied the number of integration neurons in steps of 16 neurons, from a minimum number of $4 \cdot 2^\tau$ to the maximum value 2^6 . For each combination of τ and neuron number we generated networks with the FTP algorithm, and subjected them to structural constrain (no self-connections, 4:1 Ex:In ratio, and a minimum sparsity $sp = 40\%$). We obtained 10 measurements of # *attempts* and t , from which we computed the mean and SD depicted in Fig. 5e,f.

Network construction from random transition graphs:

To construct random transition graphs that have an associated consistent system, we first constructed a matrix U_{base} , a vector Δ , and a matrix U^* of $M = 2^\tau$ rows such that the row vectors U_i^*, U_{i+1}^* and Δ were linear combinations, with indexes i being odd numbers between 1 and M . Next, we constructed matrix U in a way that ensures that each node in the graph was reachable, meaning that every node had to receive at least one edge. This is equivalent to say that every row in U^* is found at least once in U . Therefore, we set the first M rows in U equal to the U^* matrix. Then, to define the remaining rows in U , we chose $M/2$ pairs of indexes $i, i+1$, picking i values at random from the set of odd numbers between 1 and M . In this way, and unlike the transition graphs that solve an s-task, nodes could receive just one edge, or more than 2.

So far, if a row vector U_i^* appeared in matrix U more than once, then it appeared only in odd rows, or only in even rows, but not in both. This is because, indexes were ordered from 1 to M in the first half of U , and ordered in pairs of $i, i+1$ indexes in the second half. The resulting graph would be one in which any given node is reachable as the result of the presentation of either s_1 or s_2 , but not from both. In other words, if node b is reachable from node a after s_i presentation, then node b is reachable from node c only after s_i presentation, where c is any other node from which b is reachable. Following the colour code of the graph in Fig. 1b, any node receives arrows of the same colour. We wanted graphs as random as possible, so nodes reachable thought different stimuli were desired. We define these nodes as *bicoloured* nodes. In terms of indexes in matrix U , a bicolored node translates into a row vector U_i^* that appeared in matrix U in both odd and even row. For example, if we had indexes (1,2,3,4) for the first 4 rows of U , with (U_1, U_2, Δ) and (U_3, U_4, Δ) each being linearly combined, then we wanted to change this series to (1,2,2,4), or (1,2,3,1). This requires to generate new linear combinations, in particular, (U_1, U_2, U_4, Δ) will be linearly combined, for the first example, and (U_1, U_2, U_3, Δ) in the second example. Thus, we modified matrix U to generate $f_{bc}M/4$ bicolored nodes, where f_{bc} stands for 'bicolored fraction' and is a number between 0 and 1. The (nominal) maximum number of bicolored nodes is $M/4$, since we generated one node for each series of indexes i to $i+3$. Finally, we constructed matrix Z by thresholding matrix U^* , and then matrix C , which rows were in the form:

$$\begin{pmatrix} y_1 Z_{k(1)} \\ y_2 Z_{k(1)} \\ \vdots \\ y_1 Z_{k(M)} \\ y_2 Z_{k(M)} \end{pmatrix}$$

where Z_i is the i th row vector in matrix Z and vector k is a permutation of the list of integers from 1 to M .

Following the above procedure, we constructed random transition graphs that respected the linear combinations needed so that a consistent system of equations could be constructed. Given that these networks do not solve the s-task but are the minimum Frobenius norm solution to a random transition graph, we call them *F* networks. The procedure avoids index sequences like (1221), since this ordering gives a consistent system only if $\Delta = 0$, in which case stimuli cannot

be distinguished by the network. The procedure also avoids index sequences of the type (11) (one node leads to another single node through both stimuli, s_1 and s_2). If this were the case, one possibility is that s_1 and s_2 produce the same u values. Therefore, Δ is a vector of zeros and stimuli cannot be discriminated. Another possibility is that stimuli lead to different vectors u , but these vectors in turn lead to the same vector c after thresholding. This situation is possible, but would require careful selection of u values in relation to θ , and for this it was avoided.

We constructed networks that follow random transition graphs and compared their properties with the properties of networks that solve the s-task. In particular, we measured the reciprocity of the network, defined as the Spearman correlation between weights of incoming and outgoing synapses. Reciprocity was computed over matrix ω , a normalized version of the synaptic weights constructed by taking the absolute values of w and scaling them between 0 and 1. Since imposing structural constraints like Dale's principle, or sparsity, may generate many zero-valued weights, reciprocity was computed over ω values such that $\omega(i, j) \neq 0$ and $\omega(j, i) \neq 0$ for each possible pair (i, j) .

For each network generated to solve the s-task we also picked a network from its isofunction space, that is, from the set of all networks that has the same stimulus-response mapping (as described above). We refer to these networks as T networks, since they solve the s-task but they are not the minimum Frobenius norm solution. The linear mapping \mathcal{M} in eq. (10) has entries $\mathcal{M}(i, j) = r_{i,j} \max(w(i, j))$, where $r_{i,j}$ is a random number, different for each entry, sampled uniformly from the $[-1, 1]$ interval, and w is the synaptic weight matrix from which an isofunction network is desired. We defined mapping \mathcal{M} in this way to obtain isofunction networks with synaptic weight values within the range of the weights in the original network.

In addition, for each network that solves the s-task we constructed an isofunction network with structural constraints: no self-connections and Dale's principle with excitatory and inhibitory neurons in equal numbers.

All the algorithms were implemented in Matlab.

.

References:

1. Kriegeskorte, N. & Douglas, P. K. Cognitive computational neuroscience. *Nat. Neurosci.* **21**, 1148–1160 (2018).
2. Bassett, D. S., Zurn, P. & Gold, J. I. On the nature and use of models in network neuroscience. *Nat. Rev. Neurosci.* **19**, 566–578 (2018).
3. Song, H. F., Yang, G. R. & Wang, X. J. Training Excitatory-Inhibitory Recurrent Neural Networks for Cognitive Tasks: A Simple and Flexible Framework. *PLoS Comput. Biol.* **12**, 1–30 (2016).
4. Richards, B. A. *et al.* A deep learning framework for neuroscience. *Nat. Neurosci.* **22**, 1761–1770 (2019).
5. Barrett, D. G., Morcos, A. S. & Macke, J. H. Analyzing biological and artificial neural networks: challenges with opportunities for synergy? *Curr. Opin. Neurobiol.* **55**, 55–64 (2019).
6. Bengio, Y. & LeCun, Y. Scaling Learning Algorithms Towards AI. in *Large-Scale Kernel Machines* (eds. Bottou, L., Chapelle, O., DeCoste, D. & Weston, J.) 321–358 (The MIT Press, 2007).
7. Sussillo, D., Churchland, M. M., Kaufman, M. T. & Shenoy, K. V. A neural network that finds a naturalistic solution for the production of muscle activity. *Nat. Neurosci.* **18**, 1025–1033 (2015).
8. Maheswaranathan, N., Williams, A. H., Golub, M. D., Ganguli, S. & Sussillo, D. Universality and individuality in neural dynamics across large populations of recurrent networks. (2019).
9. Penrose, R. & Todd, J. A. On best approximate solutions of linear matrix equations. *Math. Proc. Cambridge Philos. Soc.* **52**, 17–19 (1956).
10. Neri, F. *Linear algebra for computational sciences and engineering. Linear Algebra for Computational Sciences and Engineering* (Springer, 2016). doi:10.1007/978-3-319-40341-0
11. Lennie, P. & Place, W. The Cost of Cortical Computation. *Curr. Biol.* **13**, 493–497 (2003).
12. Mizuseki, K. & Buzsáki, G. Preconfigured, Skewed Distribution of Firing Rates in the Hippocampus and Entorhinal Cortex. *Cell Rep.* **4**, 1010–1021 (2013).
13. Schneidman, E., Berry, M. J., Segev, R. & Bialek, W. Weak pairwise correlations imply strongly correlated network states in a neural population. *Nature* **440**, 1007–1012 (2006).
14. Strata, P. & Harvey, R. Dale's principle. *Brain Res. Bull.* **50**, 349–350 (1999).
15. Lefort, S., Tómm, C., Floyd Sarria, J. C. & Petersen, C. C. H. The Excitatory Neuronal Network of the C2 Barrel Column in Mouse Primary Somatosensory Cortex. *Neuron* **61**,

- 301–316 (2009).
16. Seeman, S. C. *et al.* Sparse recurrent excitatory connectivity in the microcircuit of the adult mouse and human cortex. *Elife* **7**, 1–27 (2018).
17. Szegedi, V. *et al.* Robust perisomatic GABAergic selfinnervation inhibits basket cells in the human and mouse supragranular neocortex. *Elife* **9**, 1–19 (2020).
18. Bekkers, J. M. Neurophysiology: Are autapses prodigal synapses? *Curr. Biol.* **8**, 52–55 (1998).
19. Calamai, P. H. & Moré, J. J. Projected gradient methods for linearly constrained problems. *Math. Program.* **39**, 93–116 (1987).
20. Song, S., Sjöström, P. J., Reigl, M., Nelson, S. & Chklovskii, D. B. Highly Nonrandom Features of Synaptic Connectivity in Local Cortical Circuits. *PLoS Biol.* **3**, e68 (2005).
21. Brunel, N. Is cortical connectivity optimized for storing information? *Nat. Neurosci.* **19**, 749–755 (2016).
22. Garlaschelli, D. & Loffredo, M. I. Patterns of link reciprocity in directed networks. *Phys. Rev. Lett.* **93**, 1–4 (2004).
23. Schulz, D. P. A., Sahani, M. & Carandini, M. Five key factors determining pairwise correlations in visual cortex. *J. Neurophysiol.* **114**, 1022–1033 (2015).
24. de la Rocha, J., Doiron, B., Shea-Brown, E., Josić, K. & Reyes, A. Correlation between neural spike trains increases with firing rate. *Nature* **448**, 802–806 (2007).
25. Constantinidis, C. & Goldman-Rakic, P. S. Correlated discharges among putative pyramidal neurons and interneurons in the primate prefrontal cortex. *J. Neurophysiol.* **88**, 3487–3497 (2002).
26. Alemi, A., Baldassi, C., Brunel, N. & Zecchina, R. A Three-Threshold Learning Rule Approaches the Maximal Capacity of Recurrent Neural Networks. *PLoS Comput. Biol.* **11**, 1–23 (2015).
27. Such, F. P. *et al.* Deep Neuroevolution: Genetic Algorithms Are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning. (2017). doi:1712.06567
28. Salimans. Evolution Strategies as a Scalable Alternative to Reinforcement Learning. *Proc. - 4th IEEE Int. Conf. Softw. Testing, Verif. Valid. Work. ICSTW 2011* 476–485 (2011). doi:10.1109/ICSTW.2011.58
29. Vinyals, O. *et al.* Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* **575**, 350–354 (2019).
30. Silver, D. *et al.* A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science (80-.).* **362**, 1140–1144 (2018).

31. Kojima, S. & Goldman-Rakic, P. S. Delay-related activity of prefrontal neurons in rhesus monkeys performing delayed response. *Brain Res.* **248**, 43–50 (1982).
32. Guo, Z. V. *et al.* Maintenance of persistent activity in a frontal thalamocortical loop. *Nature* **545**, 181–186 (2017).
33. Orhan, A. E. & Ma, W. J. A diverse range of factors affect the nature of neural representations underlying short-term memory. *Nat. Neurosci.* **22**, 275–283 (2019).
34. Murray, J. D. *et al.* Stable population coding for working memory coexists with heterogeneous neural dynamics in prefrontal cortex. *Proc. Natl. Acad. Sci. U. S. A.* **114**, 394–399 (2017).
35. Vries, M. H. De, Christiansen, M. H. & Petersson, K. M. Learning Recursion : Multiple Nested and Crossed Dependencies. *Biolinguistics* 10–35 (2011).
36. Honey, C. J., Thivierge, J. P. & Sporns, O. Can structure predict function in the human brain? *Neuroimage* **52**, 766–776 (2010).
37. Bullmore, E. & Sporns, O. Complex brain networks: Graph theoretical analysis of structural and functional systems. *Nat. Rev. Neurosci.* **10**, 186–198 (2009).
38. Vázquez-Rodríguez, B. *et al.* Gradients of structure–function tethering across neocortex. *Proc. Natl. Acad. Sci. U. S. A.* **116**, 21219–21227 (2019).
39. Kording, K. P. Bayesian statistics: Relevant for the brain? *Curr. Opin. Neurobiol.* **25**, 130–133 (2014).
40. Knill, D. C. & Pouget, A. The Bayesian brain : the role of uncertainty in neural coding and computation. *Trends Neurosci.* **27**, (2004).
41. Ellefsen, K. O., Mouret, J. B. & Clune, J. Neural Modularity Helps Organisms Evolve to Learn New Skills without Forgetting Old Skills. *PLoS Comput. Biol.* **11**, 1–24 (2015).

Acknowledgements:

This research was supported by grants from Agencia Nacional de Promoción Científica y Tecnológica (ANPCYP) PICT 2016-2145 and PICT 2017-3208) and from Universidad de Buenos Aires (UBACYT 20020170100568BA). The work of Camilo J. Mininni was supported by a postdoctoral fellowship from ANPCYT.

Author contributions:

CJM conceived the project, implemented the algorithms and wrote the initial draft. CJM and BSZ reviewed and edited the final manuscript.

Competing interests:

The authors declare no competing interests.

Materials and correspondence:

Correspondence to Camilo J. Mininni: mininni@dna.uba.ar.

Matlab codes are available upon request.