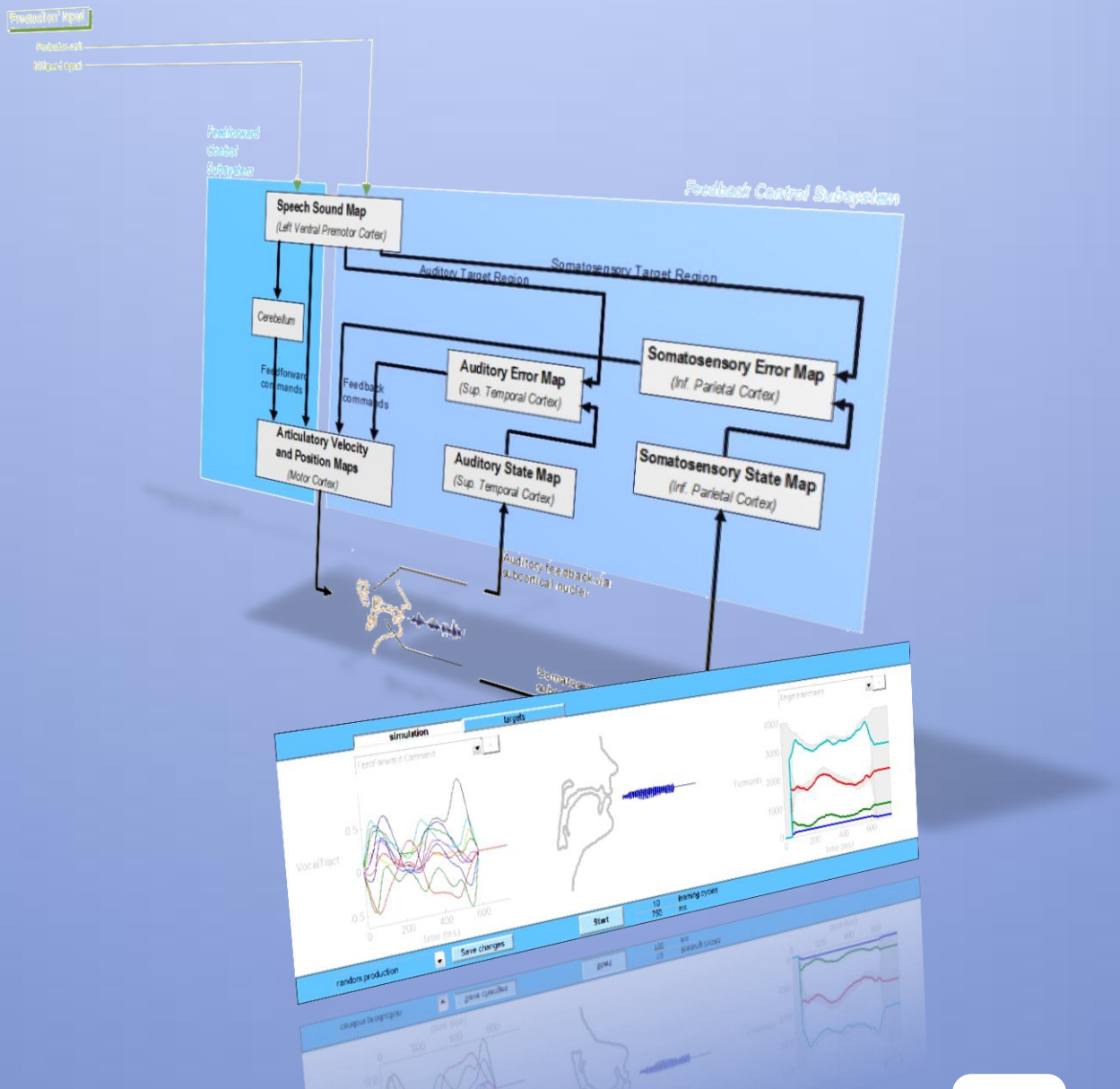

Simulink DIVA model

Manual v.2011

Contents

Installation

To install Simulink DIVA unzip the *DIVAsimulink.zip* file into any folder (e.g. /home/divasimulink/) and add this folder to the Matlab path (e.g. **addpath /home/divasimulink/**). To open the Simulink DIVA model type **open diva** in the command window. To launch the gui click on the *Production Input* link in the main DIVA system window (or type **diva_gui** in the command window).

Model overview

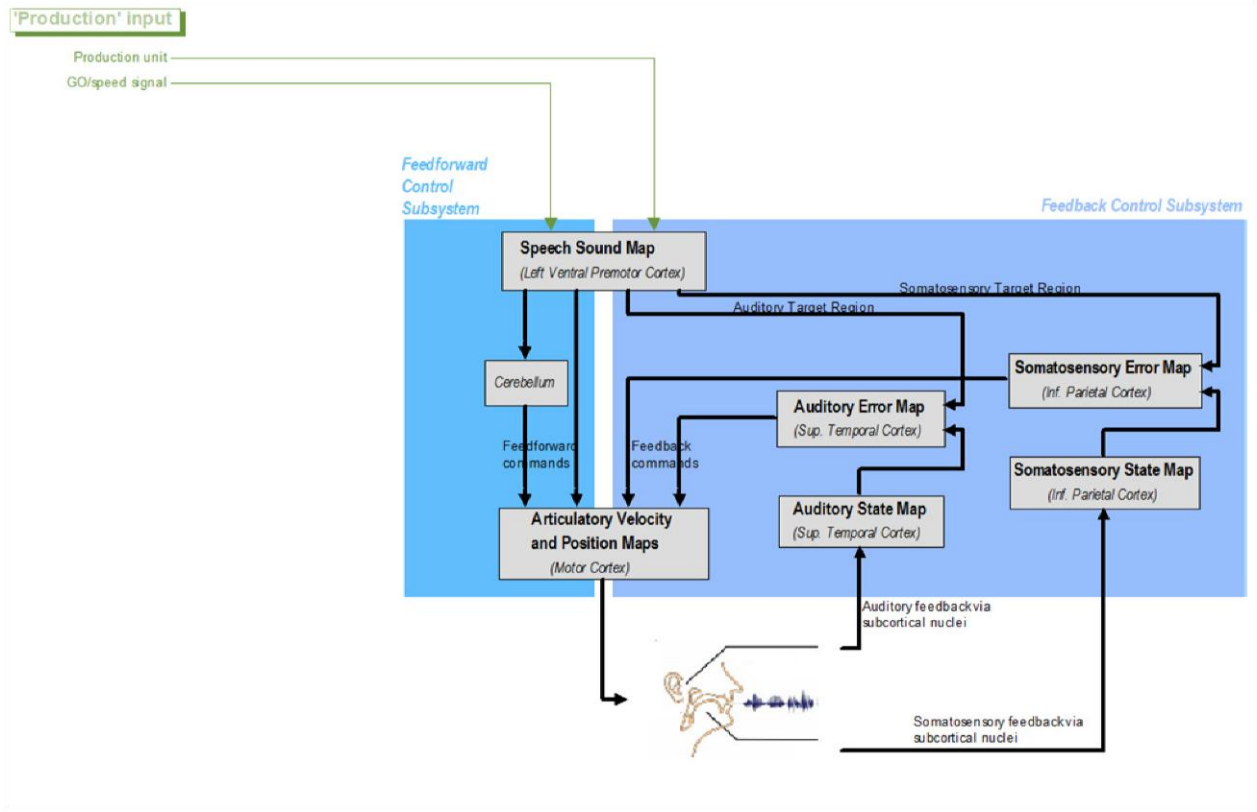

Figure 1. Main DIVA system

The Simulink DIVA model represents an implementation of the DIVA model in Matlab's Simulink environment. The model is composed of several modules (Simulink subsystems), representing different brain areas involved in speech production. To open each module simply double-click on the corresponding element in the main DIVA system window. The operation of each module is implemented through standard Simulink block operations (e.g. sums, gains, delays, convolutions, etc.), with the addition a few non-standard blocks (e.g. fixed and adaptive synaptic weights) implemented as Simulink level-2 Matlab S-functions. Standard as well as non-standard blocks typically contain block-specific parameters which can be accessed by double-clicking on each block's icon. The following sections describe in detail each Simulink subsystem in the context of the operation of the DIVA model.

Left ventral Premotor Cortex (Speech Sound Map, SSM)

SSM receives inputs from a set of *production* neurons [1], sparsely coding the range of learned speech production units (one neuron per production). When one of these inputs is active, this selects a unique set of neurons associated with that given production (through a set of fixed synaptic weights[2]). These neurons[3] sparsely code a temporal representation for each production (one neuron per timepoint per production). When SSM detects activation in its GO input signal, this generates a sweep of activation through the last set of activated neurons. The velocity of this sweep is modulated by the amplitude of the GO signal (implicitly coding the velocity of the production)[4]. The activation of these neurons is projected to different areas through differently delayed channels:

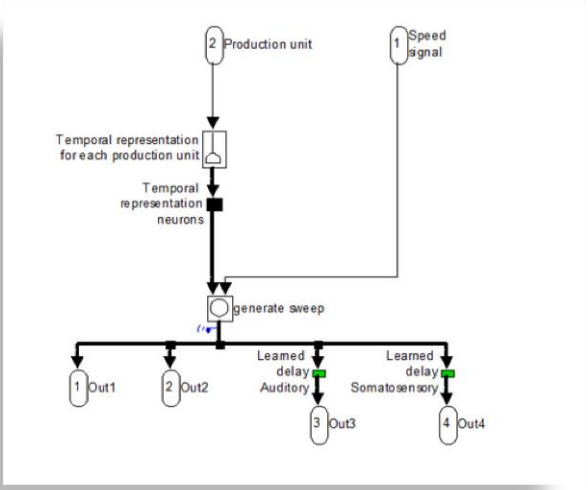

Figure 2. Speech Sound Map subsystem

a) To the *Auditory and Somatosensory Error Maps*, where the activations are delayed to coincide with the arrival in that area of the auditory/somatosensory feedback signals associated with this production[5]
b) Two outputs to Motor Cortex *Articulatory velocity and position Maps,* one without any additional delays (to initiate the corresponding motor command), and a second signal via the cerebellum implementing a learned delay[6] to coincide with the arrival in Motor Cortex of the auditory/somatosensory compensatory feedback motor command signal associated with this production (and used for learning).

[1] *Production Unit* input signal in SSM module of Simulink DIVA model
[2] *Temporal representation for each production unit* weight-block in SSM module
[3] *Temporal representation neurons* block in SSM module (note: with each neural representation we associated a *slider-gain* block in simulink, that allows the users to selectively increase/decrease the activity of these neurons to explore the model behavior under these circumstances; this can be done before a simulation starts but also interactively while the simulation is running)
[4] Implemented programmatically as *diva_sequencer* function-block in SSM module
[5] *Learned delay auditory* and *Learned delay somatosensory* delay-blocks in SSM module (fixed delay values; note: delay values are discrete, 1 delay corresponds to 5ms)
[6] *Learned delay Auditory/Somatosensory convolution-blocks in Cerebellum module (note: a convolution with a hann filter is used here instead of a fixed-delay element to implement a degree of smoothing in the error assignation; i.e. motor cortex will spread any given error-corrective motor command around the time-points close to the error-inducing production when adapting the feedforward motor command. This allows certain amount of preparatory movements to prevent an error in subsequent productions and it results in smoother learned articulatory sequences)*

Superior Temporal Cortex (Auditory State and Error Maps)

Auditory state information via subcortical nuclei arrives to superior temporal cortex after a fixed delay[7] to activate a set of neurons[8] encoding the current auditory state. These neurons encode sparsely (one neuron per feature) the pitch (F0) and formant positions (F1-F3) characterizing the current vocalization.

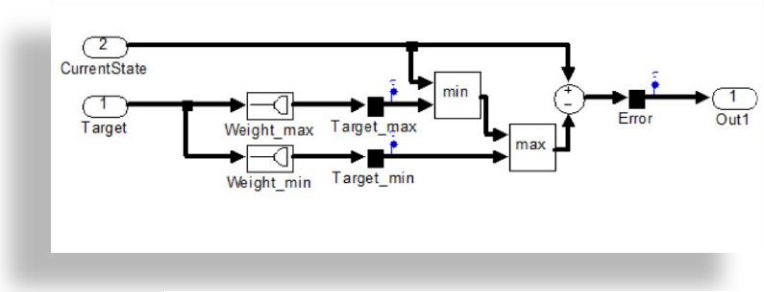

Figure 3. Auditory Error Map subsystem

Projections from SSM to Superior Temporal Cortex[9] are converted to a representation of the desired auditory target via a set of synaptic weights[10] (associating each timepoint during each production with a given auditory target region). Target regions are defined through two separate sets of neurons[11], encoding respectively the minimum and maximum value for each auditory feature. These signals are then compared[12] to the activations in the Auditory State Maps[13] to generate and auditory error signal[14], representing the distance between the auditory current state and the target region, which is then projected to Motor Cortex.

Inferior Parietal Cortex (Somatosensory State and Error Maps)

Somatosensory information is processed in the Somatosensory State and error maps following exactly the same steps as Auditory Information in the corresponding Auditory State and error maps (with the exception that the delays associated with the somatosensory feedback signal[15] are typically lower than those associated with the auditory feedback signal[16]). Somatosensory neurons[17] sparsely encode (one neuron per somatosensory feature) the vocal tract state including place of articulation[18] as well as glottal pressure and vocal fold oscillation (voicing) information. This information arrives to Inferior Parietal Cortex from subcortical nuclei, where the somatosensory error signal[19] is computed and projected to Motor Cortex.

---

[7] *Fixed delay* block in Auditory State Map module
[8] *Auditory State* block in Auditory State Map module
[9] *Target* input in Auditory Error Map module
[10] *Weight_max* and *Weight_*min weight-blocks in Auditory Error Map module
[11] *Target_max* and *Target_min* blocks in SSM module
[12] *Min*, *max*, and *sum* blocks in Auditory Error Map module
[13] *CurrentState* input in Auditory Error Map module
[14] *Error* block in Auditory Error Map module
[15] *Fixed delay* block in Somatosensory State Map module
[16] *Fixed delay* block in Auditory State Map module
[17] *CurrentState* input in Somatosensory Error Map module
[18] These somatosensory features are associated with different places of articulation. Each neuron encodes the vocal tract state at a discrete segment of the vocal tract (6 neurons, corresponding to pharyngeal, uvular, velar, palatal, alveolar-dental, and labial areas). These neurons encode the minimum distance between the interior and exterior walls of the vocal tract within each of the corresponding vocal tract segments (for open configurations this approximate distance-to-closure information, while for closed configurations this approximates force/pressure of vocal tract closure for each place of articulation).
[19] *Error* block in Somatosensory Error Map module

Motor Cortex (Articulatory Velocity and Position Maps)

Inputs from Auditory and Somatosensory error areas[20] are converted to corrective motor commands via a learned set of weights (implementing an inverse map to the Motor direction to auditory/somatosensory direction transformation for each possible motor state[21]). These motor correction signals are then combined[22] to form a *Feedback motor command.* Inputs from SSM are converted to a corresponding *feedforward motor command* through and adaptive set of synaptic weights[23] (associating each timepoint during each production with a given feedforward motor command). These feedback and feedforward signals are then combined[24], together with a small decay along the null-space direction[25] of the previous articulatory position, to define the current articulatory position[26] which is used to control the vocal tract.

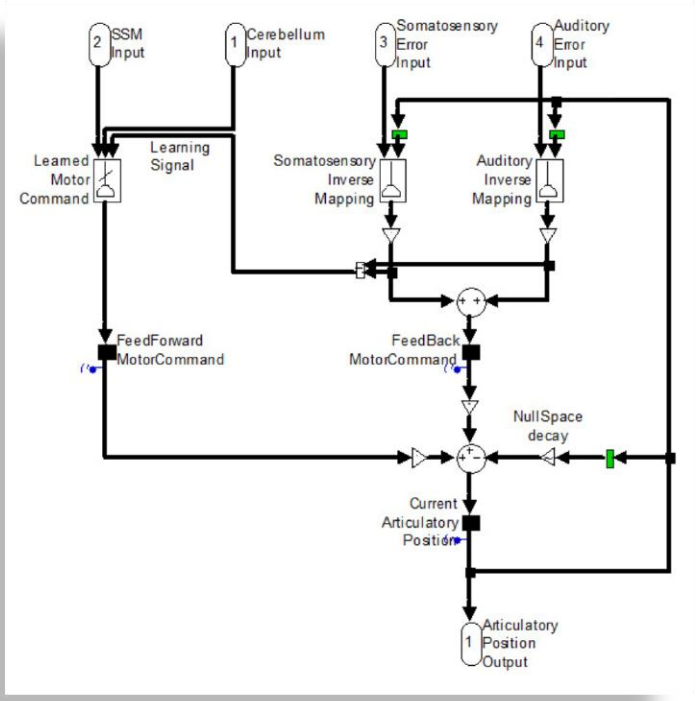

Figure 4. Motor Cortex subsystem

The auditory/somatosensory *feedback motor command* signals are also projected as a learning signal for the feedforward weights, combined with the learning signal from Cerebellum (i.e. those feedforward weights associated with the temporal units indicated by the cerebellar signal, change in the direction indicated by the feedback command signals).

The current articulatory position is also projected through a learned delay (where the activations are delayed to coincide with the arrival of the auditory/somatosensory error signals associated with this production) to inform the inverse map computations above of the articulatory state at the time of the error-generating production.

[20] *Auditory/Sotosensory Error* inputs to Motor Cortex module
[21] *Auditory/Somatosensory Inverse Mapping* blocks in Motor Cortex module. These blocks currently implement an explicit pseudoinverse estimation of the forward map Jacobians (forward maps between motor and somatosensory signals are computed explicitly using the vocal tract model, and forward maps between motor and Formant signals use a pre-computed HRBF fit to this relationship, that generalize estimated formant positions beyond closed vocal tract configurations)
[22] Gain and sum blocks in Motor Cortex module (note: gains represent scalar multiplicative factors)
[23] *Learned motor command* weight-block in Motor Cortex
[24] Gain and sum blocks in Motor Cortex module
[25] *Null-space decay* block in Motor Cortex. This blocks output, for each articulatory position input, the projection of this position vector onto the null-space of the forward map Jacobian estimated at the same articulatory position. This null space projection is explicitly estimated from the same forward maps used in the *Inverse Mapping* blocks
[26] *Current Articulatory Position* block in Motor Cortex

All of the motor representations are characterized by a set of neurons encoding sparsely (one neuron per articulatory feature) the vocal tract shape (10 articulatory dimensions), as well as glottal tension (assumed for simplicity linearly related to the resulting pitch), glottal pressure, and vocal fold oscillation (voicing) information.

To further illustrate the operation of the DIVA model Figure 5 shows the activation for the SSM production neurons, Motor Cortex's FeedForward, Feedback, and Articulatory state neurons, Superior Temporal Cortex's Auditory state and error neurons, and Inferior Parietal Cortex's Somatosensory state and error neurons, during one speech production early during speech learning (left plot, characterized by high error, and high corrective motor command signals) and after appropriate learning (right plot, characterized by low error, stronger feedforward and low feedback commands).

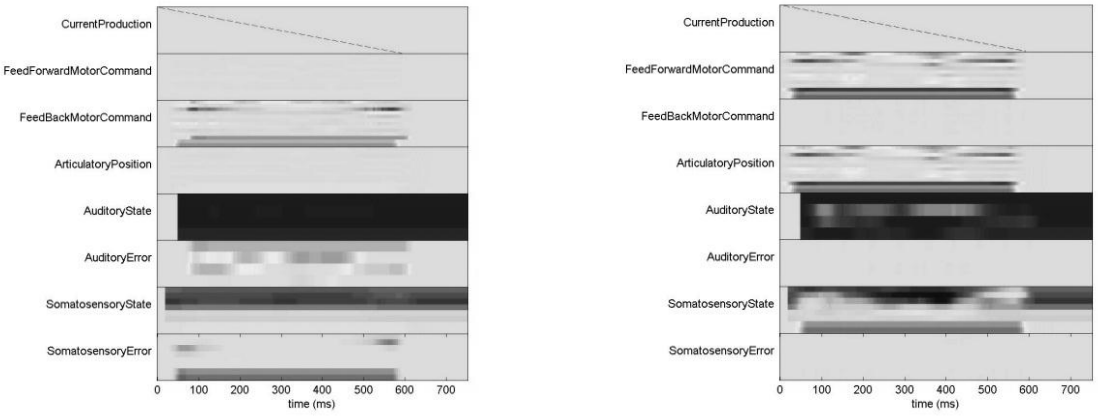

Figure 5. Example of neuron's activation profiles early (left) and late (right) during learning of a given speech production unit

Simulink model elements and parameters

Delay elements. Small green boxes in a Simulink diagram correspond to fixed-delay elements. The Simulink DIVA model is implemented using discrete fixed-step elements, where a unit delay corresponds to the fixed-step size (fundamental sample time, currently 5ms). Double-clicking on any delay element allows the user to change the associated delay time.

Gain elements. Small triangular icons in a Simulink diagram represent gain factors in the model. Double-clicking on each of these elements allows the user to modify the associated gain factors.

Slider-gain elements. Small black boxes in a Simulink diagram correspond to the activations of neural populations of interest. Double-clicking on these elements allow the user to selectively increase/decrease the activity of these neurons to explore the model behavior under these non-standard circumstances.

Synaptic weight elements. There are two types of synaptic weight elements in the Simulink DIVA model: the *standard weight* elements (e.g. in the speech sound map and error map modules) implement weights that can be modeled as a simple matrix multiplication[27]. Double-clicking on these elements allows the users to define the specific matrix characterizing each synaptic weight element[28]; the *adaptive weight* element (e.g. in the Motor Cortex module implementing the feedforward map) is just a standard weight element where the weights can be adaptively modified via additional learning inputs[29]. Double-clicking on these elements allow the user to define the learning rate as well as the starting state of the matrix characterizing each adaptive synaptic weight element.

Inverse-map elements. These elements (in the Motor Cortex module implementing the auditory and somatosensory inverse maps) implement an explicit computation of the inverse Jacobian of the vocal tract model. Double-clicking on these elements allow the user to modify: a) the portion of the output of the vocal tract model to be used in each case (Auditory or Somatosensory information); b) the Jacobian estimation step size (step-size increments in the articulatory configuration around the current articulatory position used to estimate the forward map Jacobian); and c) the Jacobian regularization factor (degree of regularization used when computing the pseudoinverse of the Jacobian).

Null-space projector element. This element (in the Motor Cortex module) implements a null-space projection of the current articulatory state. The null-space projection is defined as the operation that projects an input vector on the null-space of the Jacobian of the forward map at the current articulatory state. Double-clicking on this element allows the user to define: a) a multiplicative gain factor to be applied to the null-space projection output; b) the portion of the output of the vocal tract model to be used when estimating the forward map null-space (Auditory and/or Somatosensory information); c) the Jacobian estimation step size; and d) the Jacobian regularization factor.

---

[27] This operation is implemented through the level-2 matlab S-function *diva_weights.m*. While this operation could have been defined directly using standard Simulink's matrix multiplication block, this implementation was chosen to facilitate future generalizations implementing more sophisticated synaptic weight operations.

[28] Each .mat file should contain a variable named **W** with as many rows as the dimensionality of the inputs, and as many columns as the dimensionality of the outputs. For any given input **x** the output will be $\mathbf{y}=\mathbf{W}^t\cdot\mathbf{x}$. Optionally the .mat can also include one additional vector variable named **W0** characterizing a bias element (the output **y** in the case of $\mathbf{x}=0$).

[29] The adaptive-weight element contains three inputs, **x**, **a**, and **b**. The output of the weight operations is, as before, $\mathbf{y}=\mathbf{W}^t\cdot\mathbf{x}$. In addition, the weights are incremented at each time-point by a factor proportional to $\Delta\mathbf{W}=\mathbf{a}\cdot\mathbf{b}^t$. This functionality is implemented in the level-2 matlab S-function *diva_weightsadaptive.m*.

Simulink DIVA gui

The Simulink DIVA gui allows users to easily prepare a simulation of the DIVA model and visually explore the simulation results. The Simulink DIVA system requires the definition of the Production Unit and GO signal inputs, as well as the specification of the weight matrices characterizing the model synaptic weights. While advanced users might wish to define these parameters manually in the context or a larger Simulink model or simulation (see **Advance use** section), the gui allows users to: a) run a simulation intended to produce a well-learned speech production unit; and b) defining new speech production units (or editing a existing one) and running a set of simulations to train the DIVA system to produce this new unit.

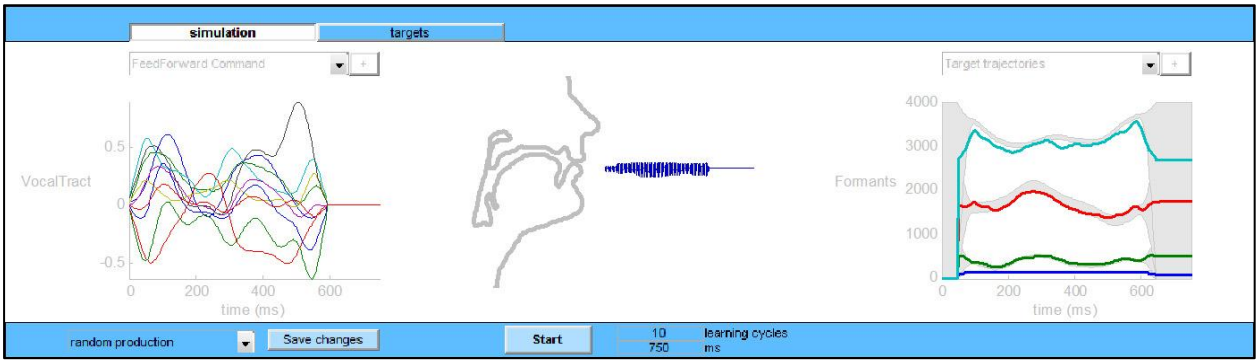

Figure 6. Simulink DIVA gui *simulation* tab

Producing a well-learned speech-production unit

Selecting from the bottom-left dropdown list the desired speech-production unit will load the desired target. Click the *Start* button to start a simulation and have DIVA produce the corresponding speech sound. While the simulation is running the center plot will display the evolution of the vocal tract configuration over time. When the simulation ends (this will typically take slightly longer than the actual time-length of the desired production) the gui will display by default the timeseries associated with the FeedForward motor command (left plot) and auditory state and targets (right plot), and it will also play the corresponding sound through the speakers. Users can select to view Motor state, FeedBack commands, or Feedforward commands using the dropdown list above the left plot. Clicking on the associated '+' icon allows the user to select which portion of the motor representation to display (vocal tract shape parameters -10 dimensions-, glottal tension, pressure, and/or voicing). Users can also select to view Target trajectories or Target error signals using the dropdown list above the right plot. Clicking on the associated '+' icon allows the user to select which portion of the auditory/somatosensory representation to display (Formants –F0 to F3-, place of articulation -6 dimensions-, glottal pressure, and/or glottal voicing).

Defining a new or editing a existing speech-production unit

Selecting the *targets* tab in the upper section allows the user to define new speech production targets and edit existing ones. Selecting from the bottom-left dropdown list the desired speech-production unit will load the desired target. Each target can be characterized by timeseries defining the target region for each auditory and somatosensory feature. In order to simplify the definition of these timeseries, the gui allows the user to define a series of control points and the timeseries are generated from these control points through nearest, linear, or spline interpolation. When defining a new speech production target users may use as a starting definition a default target configuration corresponding to a production in a resting configuration of the vocal tract (select *new@default* in the bottom-left dropdown list), or they may use a random target configuration corresponding to an open vocal tract configuration and three random control positions in F0-F4 space (select *new@random* in the bottom-left dropdown list). Selecting one or several fields in the *Target field(s)* list allows the user to display the

control points and associated timeseries for the corresponding auditory/somatosensory features. Users can edit the number and position of each control point (*Control points (ms)* dialog), and the associated minimum and maximum target values for each feature and control point (*Minimum/Maximum value* dialogs; each dialog contains a list of numbers with as many rows as target fields selected, and as many columns as control points defined). After editing these values click *Save changes* (and optionally select whether to delete any learning already computed on this target –*Reset learned feedforward sequence* checkbox) and proceed to the *simulation* tab to run a set of simulations to train the DIVA system to produce this new unit.

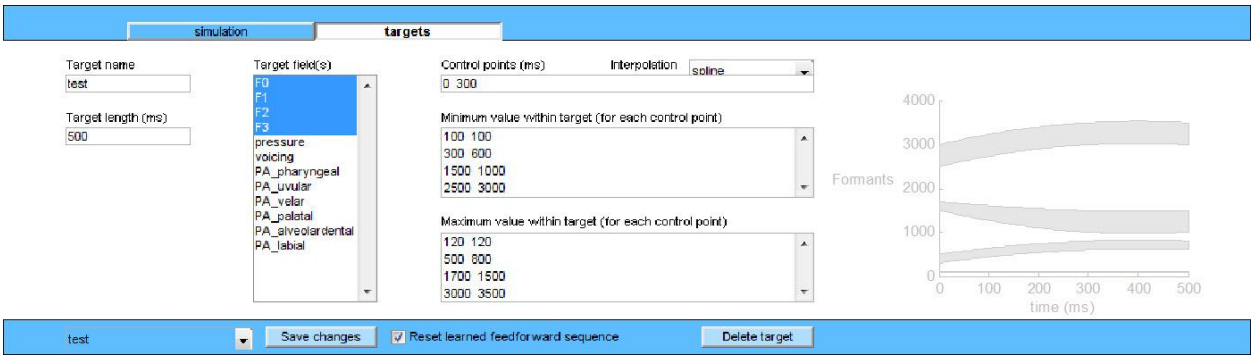

Figure 7. Simulink DIVA gui *targets* tab

Advance use

Simulink DIVA model input/outputs

Inputs to the Simulink DIVA model consist of a *Production unit* signal (sparse representation selecting one speech production unit) and a *GO/speed* signal. The *GO/speed signal* is set by default to the value of 1 during the entire simulation (so a production is initiated as soon as an input from the *Production unit* selects the desired speech target). The *Production unit* input is loaded from the base workspace (variable *Target_production*). In addition to these inputs, the simulation needs to know the initial states of all of the synaptic weights in the model (both the standard and the adaptive weights). These are defined through a series of .mat files (one .mat file per weight element). The corresponding filenames are defined explicitly as parameters for each synaptic weight block within the model. The dimensionality of all of the signals in the Simulink model are flexible (e.g. the number of neurons in each representation is not fixed by the Simulink model) and they are inherited from the dimensionality of the inputs/outputs of the vocal tract model (which are defined by the vocal tract block itself) and the dimensionality of the weight matrices associated with each synaptic weight block. As long as all of these dimensions are self-consistent, the simulation will be able to derive the correct dimensionality for each of the signals in the Simulink model. When defining new blocks for the DIVA model we recommend using inherited sample times and inherited dimensionality of inputs whenever possible in order to maintain the flexibility of the resulting Simulink system.

The Simulink DIVA model logs all the relevant neural signals timeseries after each simulation. These timeseries can be further explored through the global variable DIVA_x.logs

Simulation step-size (fundamental sample time)

The Simulink implementation of DIVA is defined as a discrete-time process. The simulation fundamental sample time (currently 5ms, 200Hz) is explicitly defined by the sample-time of the *Production unit* input to the model (the rest of the model use inherited sample times). It is sometimes useful to use different step-size increments (e.g. use larger step-size increments to speed up a simulation computation time, or use smaller step-size increments to increase the temporal resolution of the results). It is possible to change the simulation step-size simply by defining a different sample-time in the *Productions unit* Source block (e.g. to use 1ms steps, simply change the sample-time value from .005 to .001). Since some of the model parameters are defined in step-size (τ) or related units (e.g. velocities defined in τ^{-1} units) some of the models parameters need to be rescaled when changing the simulation fundamental time. For example, if changing the simulation sample time from 5ms to 1ms , the following elements would need to be scaled: a) **delay elements** (all subsystems) need to be multiplied by 5 to maintain the same absolute delays in ms units; b) **FIR filters** (cerebellum subsystem) need to be rescaled to filter lengths 5-times the original size to maintain the same filter support in ms units (e.g. use diva_hanning(0,20,20) instead of diva_hanning(0,4,4)); c) **learning rates** (motor cortex subsystem, adaptive weights) need to be divided by 5 to maintain the same absolute speed in ms^{-1} units; and d) **GO/speed** signal (main DIVA system) needs to be divided by 5 to maintain the same absolute speed in ms^{-1} units.

Interpretation of neural activations

Both Motor and Somatosensory/Auditory representations are encoded as vectors of real numbers. Their values are scaled so that they will approximately range between -1 and 1. Motor representations encode the vocal tract shape (10 dimensions obtained from a principal component decomposition of midsagittal MRI vocal tract profiles, see Figure 8), the glottal tension (1 dimension, controlling the resulting production pitch), glottal pressure (1 dimension, controlling the resulting production amplitude), and vocal fold oscillation information (1 dimension, controlling voicing). Auditory representations encode formant positions (F0 to F3). These are defined in Hz through the gui target definition scripts but they are scaled to an approximate [-1,1] range for the corresponding neural

activations in the DIVA model (see *Scale* parameters in *diva_vocaltract.m* script). Activations are then rescaled back to the original Hz units by the gui when displaying the simulation results. Somatosensory representations encode vocal tract state (place of articulation, 6 dimensions; see Figure 9), glottal pressure, and vocal fold oscillation information. These values are already naturally scaled to a [-1,1] range. Positive numbers for place of articulation variables indicate vocal tract contact/closure at the corresponding vocal tract segment, while negative numbers indicate manner of articulation (approximately distance to closure). Zero or negative values for glottal pressure indicate the absence of any glottal pressure. Positive/negative numbers for vocal fold oscillation indicate voiced/unvoiced glottal sources.

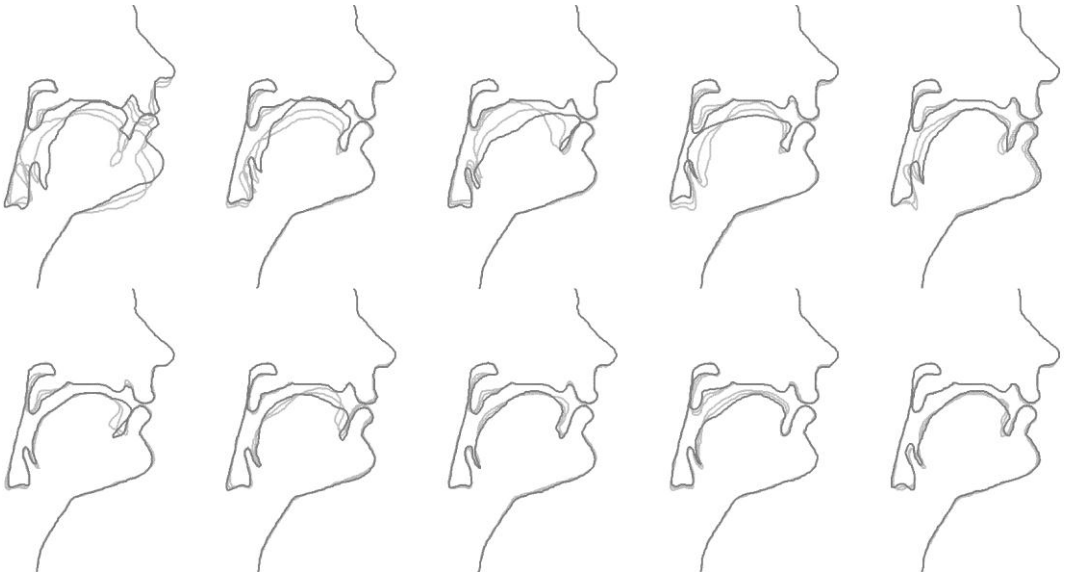

Figure 8. Motor representation. Articulatory dimensions controlling vocal tract shape (10 dimensions, from left to right and top to bottom, representing the first 10 dimensions of Motor neural representations in the Simulink DIVA model)

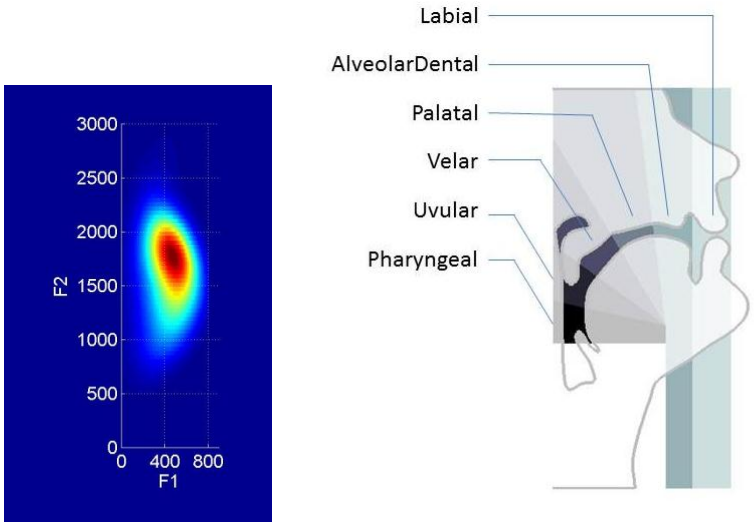

Figure 9. Auditory/Somatosensory representation. Left: Formant space (range of F1-F2 formant positions achievable by the vocal tract model, representing two dimensions of Auditory neural representations in the Simulink DIVA model). Right: Place of articulation (6-dimensions, from bottom to top, representing the first 6 dimensions of Somatosensory neural representations in the Simulink DIVA model)

Editing the Simulink model

Like any Simulink model, it is very simple and intuitive to edit the Simulink DIVA model to add new elements or modify its behavior. As an example the following steps would be needed to model an auditory-perturbation experiment (where the subject auditory feedback is modified on real-time while producing speech). We will model this by adding an intermediate block between the Auditory output of the vocal tract block and the input of the Auditory state map, that will add a step perturbation to the Auditory output (formant positions) of the vocal tract before feeding it back to the Superior temporal cortex block.

1. On the main DIVA window, click on *View->Library Browser* to display the library of Simulink standard blocks, and then drag and drop the *Sum* (from the *Math Operations* library) and *Step* (from the *Sources* library) onto the main DIVA window
2. Disconnect the Auditory output of the vocal tract (right-click on the arrow and select *delete*), and reconnect this output to one of the inputs of the *Sum* block (left-click on the small arrow at the now empty Auditory output of the vocal tract block and drag it to one of the inputs of the *Sum* block). Then connect the output of the *Step* block to the second input of the *Sum* block, and the output of the *Sum* block to the input of the Auditory state map block
3. Now we only need to define the size and time of the perturbations added to the auditory signal. Double-click on the *Step* block and enter the time of the onset of the perturbation (in seconds) in the *Step time* field (e.g. 0.1). In the *Final value* field we need to enter the perturbation size as a four-dimensional vector (for F0-F3 values). For example we can enter [0,0,0.1,0] to signify a perturbation of the second formant with size 150Hz (note: the neural representations are scaled so that all have approximately the same range, the scaling factor for F2 is 1500Hz, so a value of 0.1 corresponds to 150Hz change)
4. Last we can tide up slightly the resulting model. Select the *Sum* and *Step* blocks, then right-click and select *Create subsystem*. This will create a subsystem for the perturbation elements that we just created hiding it from the main DIVA window. Rename the new subsystem to *Auditory perturbation.* Then right-click on the new subsystem and select *Mask subsystem* to create a custom mask for this subsystem. In the *Icon* tab, fill the *drawing command* field to your liking (e.g. *plot([1,2,2,3,3,4],[0,1,0,1,0,1])* to create an icon for this subsystem). In addition in the *Parameters* tab we can add two parameters to create a set of custom properties of this new perturbation block: a) for the first parameter enter *Perturbation size (F0-F3 in Hz)* in the *prompt* field, *perturbationsize* in the *variable* field, *edit* in the *type* field, and select *evaluate* and *non-tunable*; for the second parameter enter *Perturbation time (in ms)* in the *prompt* field, *perturbationtime* in the *variable* field, *edit* in the *type* field, and select *evaluate* and *non-tunable*. Now you can double-click on the *perturbation* block and enter the desired perturbation time and size (e.g. 100 and [0,0,150,0] for the same values as before). Last right-click on the *perturbation* block and select *Look under Mask*, the double-click on the *Step* block, and enter *perturbationtime/1000* in the *step time* field (instead of .1), and *perturbationsize./[100,500,1500,3000]* in the *final value* field (instead of [0,0,.1,0]).

The resulting model is now ready (this should look similar to the diva_perturbation.mdl in the DIVAsimulink release). Users can double-click on the new *Auditory perturbation* icon on the main DIVA window and enter the desired perturbation parameters (perturbation time in ms, and perturbation size in Hz). Running a new simulation you will observe how the Auditory trajectories are affected by the perturbation and how the DIVA system adapts to this effect (a few milliseconds after the perturbation and also over several learning cycles)

Release notes and future directions

These are technical aspects of the current implementation that are active areas of research and may change in future releases.

1) role of cerebellum. We assumed that the 'direct' projection between premotor cortex and motor cortex informs the motor cortex of what portion of the learned motor command should be executed at each time point, and that the 'indirect' projection through the cerebellum informs the motor cortex of what portion of the feedforward command should be adapted/learned at each time point (this latter projection needs to be delayed to match the arrival of the corrective motor commands -see point 2 below-). This seemed a reasonable assumption matching the involvement of cerebellum in learning the feedforward commands, but needs further work to ensure this reflected well the involvement of the cerebellum.

2) temporal delays. We included 'intrinsic' small delays (5ms) for the cortico-cortical lines between premotor cortex and motor cortex as well as between auditory/somatosensory areas and motor cortex, and longer delays (50ms) for the auditory vocal tract channel (time between motor commands and auditory input to superior temporal cortex) and also (20ms) for the somatosensory vocal tract channel (time between motor commands and somatosensory input to inferior parietal cortex). The rest of the delays in the model correspond to somewhat 'learned' or 'necessary' delays: a) the delays between premotor cortex and auditory/somatosensory areas (55ms and 25ms, respectively) are set to make the auditory/somatosensory expectation signals arrive at the error maps at the same time that the corresponding auditory/somatosensory state signals (so that the error signals are computed correctly); b) the delays between premotor cortex and motor cortex (the one passing through the cerebellum) are similarly set (55ms and 25ms) to make the learning signals arrive at the motor cortex at the same time that the corresponding feedback corrective command signal (so that the correct portion of the feedforward command are adapted); and c) the delays within motor cortex (possibly passing through cerebellum but I was not sure what you thoughts of this) between the articulatory commands and the inverse-maps are also set (55ms and 25ms) to make the articulatory position signal arrive at the inverse-maps at the same time than the corresponding error signals (so that the inverse projection is computed based on the articulatory configuration at the time of the error-generating production). These latter delays are necessary for the correct behavior of the model but they could be implemented in many different ways (e.g. we could differentially delay the auditory/somatosensory areas projection to motor cortex to make the somatosensory and auditory error signals arrive at the same time despite different 'intrinsic' delays, so the the rest of the 'learned' delays in/to motor cortex need not differentiate between the auditory/somatosensory signals). Further work would be needed to better flush out the role and particular implementation of these learned delays in the DIVA model.

3) null-space projection. We included the decay along the null-space direction (as typically specified in the diva model) as part of the update of the current articulatory position. We also tried implementing this as part of the inverse-map computation instead (so that we would not need to model the null-space computation explicitly) but the original implementation seemed to behave better in the presence of smoothing and non-linearities in the forward maps (the latter implementation would still converge but not to a zero-error state).

4) smoothing. We included a small degree of smoothing in the feedforward learning simply by convolving the learning signals from premotor cortex to motor cortex with a small hann window (this has the effect of spreading the corrective motor commands in time when learning the feedforward command, which increase the robustness of the learning and produces smoother trajectories). In the curent implementation this step is not terribly important since the forward maps are already very stable and noise-free, but this seemed like a natural way to increase the robustness of the learning in a more general scenario.

Contact alfnie@gmail.com for questions/feedback about this Simulink implementation of the DIVA model