# Readme

February 14, 2026

The algorithm is divided into 4 files: 1) the *run_hessian_solver.m* file that runs the main code and contains the data generating functions; 2) the *PolynomialG.m* file that contains the class for the recurrence function $G$; 3) the *PolynomialPsi.m* file that contains the class for the test function $\psi$ and 4) the *HessianSolver.m* file that contains the class that runs the main algorithm.

The PolynomialG class has two functions: 1) *forward_trajectory_only*, which computes $G = G(h_t, x_t; w)$ by running the recurrence and 2) *forward_second_order* that computes both $G$ and its Jacobian and Hessian with respect to the parameters $w$, using the formulas

$$J_t = \frac{\partial G}{\partial w} + \frac{\partial G}{\partial h_{t-1}} J_{t-1}$$

(in the code `Jh = df_dw + df_dh * Jh_prev` in line 196)

and

$$H_t = \frac{\partial^2 G}{\partial w^2} + \left( \frac{\partial^2 G}{\partial w \partial h_{t-1}} J_{t-1}^T \right) + \left( \frac{\partial^2 G}{\partial w \partial h_{t-1}} J_{t-1}^T \right)^T + \frac{\partial^2 G}{\partial h_{t-1}^2} J_{t-1} J_{t-1}^T + \frac{\partial G}{\partial h_{t-1}} H_{t-1}$$

(in the code `Hh = g_ww + out_gwh_J + out_gwh_J' + g_hh*out_JJ + dg_dh*Hh_prev` in line 250).

Notice too that the coefficients of $G$ are *tanh*-squashed and the powers $x^i, h^i$ are normalized with the factorial $i!$ to prevent explosion.

The same squashing and normalization is applied to the coefficients and the powers of $y^i$ in the PolynomialPsi class. Just like PolynomialG, PolynomialPsi has two functions: 1) *forward* returns only $\psi(Y)$ and 2) *get_derivatives* returns both $\psi(Y)$ and its first and second derivatives with respect to both $y$ and the parameters $w$, using that

$$\psi(y; w) = \sum_{i=0}^{D} \frac{tanh(w_i)}{i!} y^i$$

implies

$$\psi_{y,w_i} = (1 - tanh(w_i)^2)(i - 1!)^{-1} y^{i-1}$$
$$\psi_{w_i,w_j} = \delta_{ij}(-2) tanh(w_i)(1 - tanh(w_i)^2)(i!)^{-1} y^i$$

Finally, the *update_step* in *Hessiansolver.m* implements and solves the system $H(dw, dy) = -D$ for

$$H = \begin{pmatrix} I_{d_w} - \eta_w \left( L_{ww} - ({L_{yy}}^{-1} L_{yw})' L_{yw} \right) & -\eta_w \left( L_{wy} - ({L_{yy}}^{-1} L_{yw})' L_{yy} \right) \\ \eta_y L_{yw} & I_{d_y} + \eta_y L_{yy} \end{pmatrix}$$

and

$$D = \begin{pmatrix} -\eta_w \ \left( \nabla_z L - ({L_{yy}}^{-1} L_{yz})' \nabla_y L \right) \\ \eta_y \ \nabla_y L \end{pmatrix}$$