

Neural Boltzmann Equations

Jonas Spinner, Jack Shergold

arXiv:2608.23022

EuCAIFCon 2026
Heidelberg



Alexander von
HUMBOLDT
STIFTUNG

Neural Boltzmann Equations

J. Spinner¹, J. D. Shergold²

¹Institute for Particle Physics Phenomenology, Durham University, United Kingdom

²Department of Mathematical Sciences, University of Liverpool, United Kingdom



Durham
University

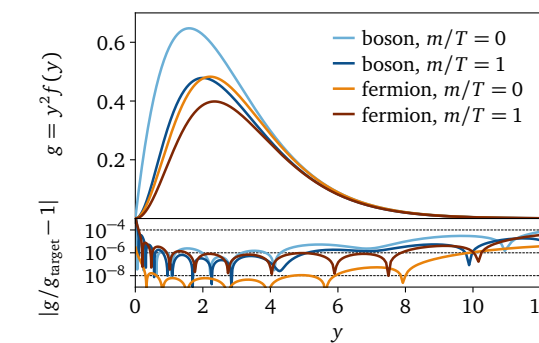
Neural distribution functions

- Factorise $f(y)$ into normalisation η and normalised shape $p(y)$

$$f(y) = \frac{4\pi^2 E}{y^2} \eta p(y), \quad \eta = \int dy \frac{y^2}{4\pi^2 E} f(y), \quad \int dy p(y) = 1$$
- Density $p(y)$ as gamma mixture in the kinetic energy ε

$$p(y) = \frac{y}{ET} \sum_{m=1}^M w_m \gamma(\varepsilon | \alpha_m, \lambda_m), \quad \gamma(\varepsilon | \alpha, \lambda) = \frac{\lambda}{\Gamma(\alpha)} (\lambda \varepsilon)^{\alpha-1} e^{-\lambda \varepsilon}, \quad \varepsilon = \frac{\sqrt{y^2 + m^2} - m}{T}$$
- Ensure $f \leq 1$ for fermions with an unconstrained f_0 as $f = f_0 / (1 + f_0)$
- Predict parameters ω with NN conditioned on theory ξ

$$\omega = \{\eta, (w_m, \alpha_m, \lambda_m)_{m=1}^M\} = \text{NN}(\xi; \theta)$$

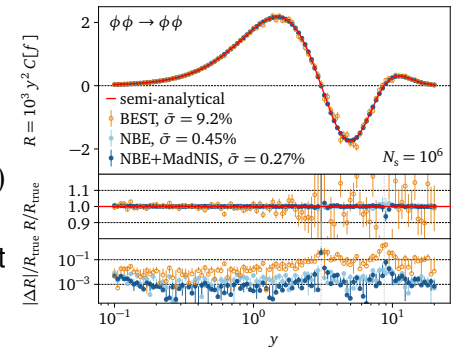


Collision operator

- MC integral over all other particles

$$C_a[f] = \frac{1}{2E_a} \sum_k \int d\Phi_k |\mathcal{M}_k|^2 \Lambda_k[f]$$

$$\Lambda_k[f] = \prod_{b \in k_{\text{out}}} f_b \prod_{c \in k_{\text{in}}} (1 \pm f_c) \quad (\text{in} \leftrightarrow \text{out})$$
- MadSpace [2] mappings, $f(y)$ sampling, MadNIS flow refinement
- NBE 1000× more sample efficient than BEST baseline [3]



Algorithm 1 Phase space generation for a process with m incoming and $n - m$ outgoing particles.
Require: observed momentum y_i , condition ξ , masses $m_1, \dots, m_n$, proposal densities $p_1, \dots, p_n$
 1: $\hat{y}_i \leftarrow y_i \xi$ → observed leg, pinned to a fixed axis $\hat{z}$
 2: $y_k \sim p_k(\cdot | \xi)$, $\hat{\Omega}_k \sim \mathcal{U}(S^2)$ for $k = 2, \dots, m$ → sample initial legs
 3: $y_{m+1} \sim \mathcal{U}([0, 1]^{3(n-m)})$
 4: **If** neural importance sampling **then**
 5: $u_i, l_{\text{out}} \leftarrow \text{ToUnitCubic}(y_1, \dots, y_m, \hat{\Omega}_1, \dots, \hat{\Omega}_m, u_{\text{out}})$
 6: $u_i, l_{\text{out}} \leftarrow \text{Flow}(u_i | y_i, \xi)$ → MadNIS refinement
 7: $(y_1, \dots, y_m, \hat{\Omega}_1, \dots, \hat{\Omega}_m, u_{\text{out}}), l_{\text{out}} \leftarrow \text{ToUnitCubic}^{-1}(u)$
 8: **else**
 9: $l_{\text{out}}, l_{\text{out}}, l_{\text{out}} \leftarrow 1$
 10: $\hat{y}_i \sim y_i \hat{\Omega}_i$ for $k = 2, \dots, m$ → build initial state
 11: $\sqrt{s} = \left((\sum_{i=1}^m E_i)^2 - |\sum_{i=1}^m \hat{y}_i|^2 \right)^{1/2}$ with $E_i = (y_i^2 + m_i^2)^{1/2}$
 12: $\hat{y}_{m+1}, \dots, \hat{y}_n, l_{\text{out}} \leftarrow \text{MadSpace}(u_{\text{out}}; \sqrt{s}, m_{m+1}, \dots, m_n)$ → build final state
 13: $w_{\text{in}} \leftarrow l_{\text{out}} l_{\text{out}} l_{\text{out}} \prod_{i=1}^m \frac{1}{p_i(y_i)}$ → importance weight
 14: **return** $(\hat{y}_1, \dots, \hat{y}_n, w_{\text{in}})$

Boltzmann equations

- Solve for particle distribution function $f_a(y)$ evolution in expanding universe
- Comoving quantities: time x , momentum y
- Examples: dark matter production, baryogenesis/leptogenesis, neutrino decoupling

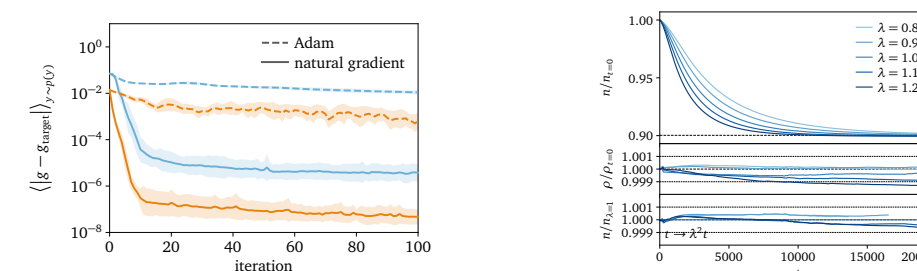
$$\frac{\partial f_a(y)}{\partial \log x} = \frac{1}{H} C_a[f]$$

Challenges

- Increase overall numerical precision
- Advanced collision integrals (higher-multiplicity processes, NLO)
- Flexible and easy to use framework
- Theory parameter conditioned training for efficient scans

Boltzmann equation solver

- Fixed-point iteration with stiffness-aware target
- Natural gradient updates include function space geometry



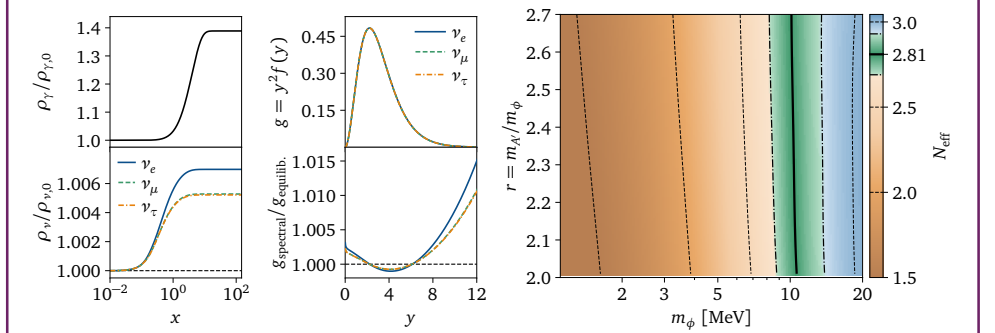
Algorithm 2 Boltzmann solver step $\log x_n \rightarrow \log x_{n+1}$ fixed-point iteration
Require: condition prior $p_{\text{prior}}(\xi)$, frozen previous solutions $\theta_n, \theta_{n-1}, \dots$
 1: $\theta_{n+1} \leftarrow \theta_n$ → warm-start from previous step
 2: **repeat**
 3: $\xi_i \sim p_{\text{prior}}(\xi)$ for $i = 1, \dots, B$ → sample conditions
 4: $\xi \leftarrow (\xi_1, \dots, \xi_B)$ → batched evaluation
 5: $\omega_i \leftarrow \text{NN}(\xi; \theta)$ for $i = 1, \dots, n$ → cache distribution function parameters
 6: **for** species $a = 1, \dots, N$ **do** → update each species independently
 7: $y \sim p(y | \omega_{a+1}, \omega_a)$ → sample test momenta
 8: $s(y), \kappa(y) \leftarrow$ evaluate collision integral
 9: $A(y), \gamma \leftarrow$ evaluate scheme coefficients
 10: $\hat{g}_{\text{approx}}(y) \leftarrow T(g(y; \omega_{a+1}, \omega_a), s(y), \kappa(y), A(y), \gamma)$ → fixed-point target
 11: $r_i \leftarrow g(y; \omega_{a+1}, \omega_a) - \hat{g}_{\text{approx}}(y)$
 12: $w \leftarrow [r(y; \omega_{a+1}, \omega_a) g(y; \omega_{a+1}, \omega_a)]^{-1}$
 13: $J \leftarrow \frac{\partial \hat{g}_{\text{approx}}}{\partial \omega}$ using autodiff
 14: $W \leftarrow \text{diag}(w), F \leftarrow J^T W J$ → natural-gradient step
 15: $\theta_{n+1,2} \leftarrow \theta_{n+1,2} - F^{-1} J^T W r$
 16: **until** convergence criteria satisfied

Example: neutrino decoupling

- Precision Standard Model calculation of N_{eff} , the effective number of relativistic neutrino degrees of freedom
- Excellent agreement with the literature result $N_{\text{eff}} = 3.0440 \pm 0.0001$ [4]

Description	NBE	Literature
Instantaneous decoupling	—	3 (exact)
Non-instantaneous	3.0340 ± 0.0001	3.0350 ± 0.0010
+ finite-temperature QED	3.0434 ± 0.0001	3.0443 ± 0.0010
+ spectral distortions	3.0434 ± 0.0001	3.0434 ± 0.0001
+ flavour oscillations	3.0439 ± 0.0001	3.0440 ± 0.0001

- Conditioned NBE scans the parameter space of a BSM model [5] that can resolve the 2σ tension with $N_{\text{eff}} = 2.81 \pm 0.12$ [6]



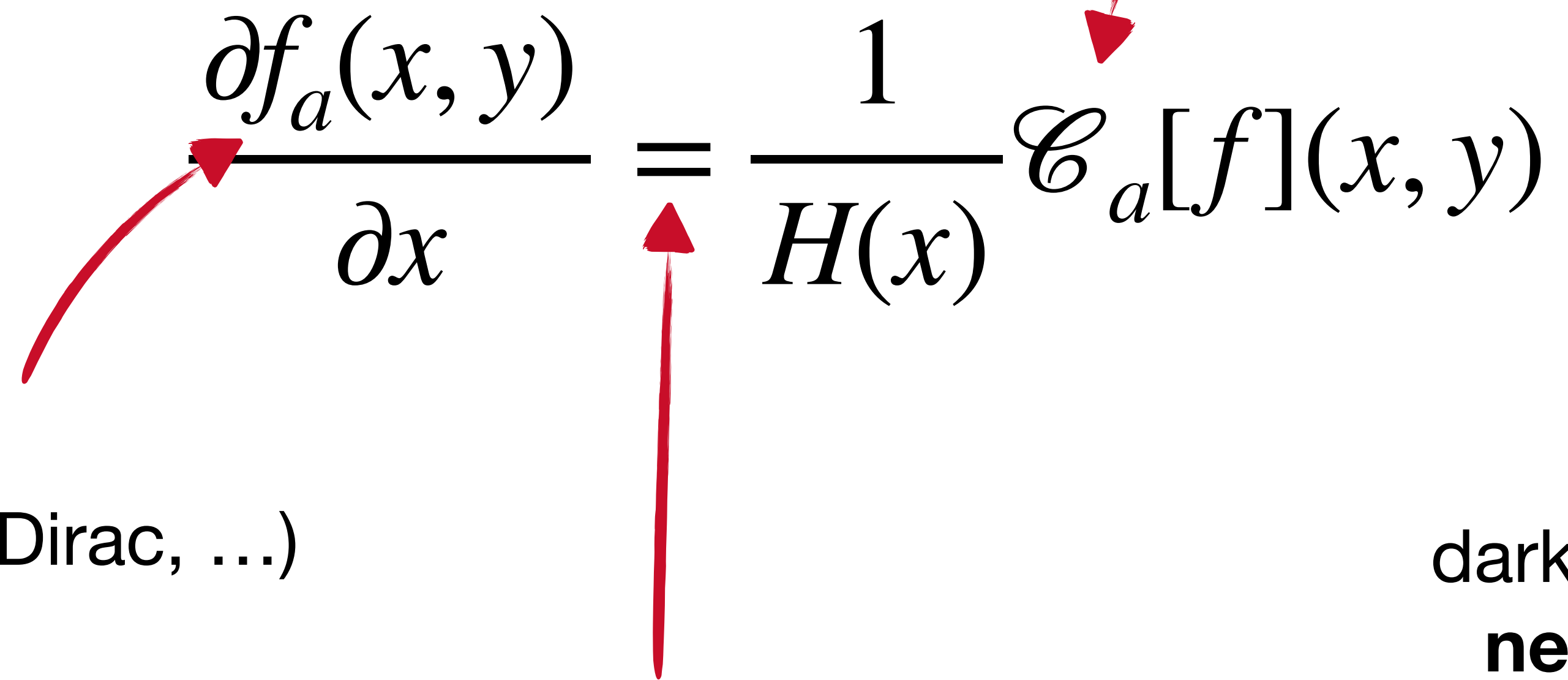
References

- Spinner, Shergold, "Neural Boltzmann Equations." *arXiv:2608*. (2026).
- Heimel, Mattelaer, Winterhalder, "MadSpace – Event Generation for the Era of GPUs and ML." *arXiv:2602.06895* (2026).
- Yoon, "Boltzmann equation solver for thermalization (BEST)." *Comput. Phys. Commun.* 327, 110295 (2026), *arXiv:2603.28848*.
- Froustey, Pitrou, Volpe, "Neutrino decoupling including flavour oscillations and primordial nucleosynthesis." *JCAP* 12, 015 (2020), *arXiv:2008.01074*.
- Escudero, Ovchinnikov, Weiner, "What does it take to have $N_{\text{eff}} < 3$ at CMB times?" *arXiv:2603.22391* (2026).
- Louis et al. (ACT), "DR6 power spectra, likelihoods and Λ CDM parameters." *JCAP* 11, 062 (2025), *arXiv:2503.14452*.

How to simulate the early universe

Comoving quantities:
time x , momentum y

Collision operator
(phase space integral)

$$\frac{\partial f_a(x, y)}{\partial x} = \frac{1}{H(x)} \mathcal{C}_a[f](x, y)$$


Distribution function f_a
(Bose-Einstein, Fermi-Dirac, ...)

Match particle number changes
from expansion and collisions

Applications:
dark matter production,
neutrino decoupling,
leptogenesis/baryogenesis

Computational challenges

Classical approach

Evolve $f_a(x, y)$ on
fixed momentum grid

Rewrite collision operator
and use quadrature methods

Parameter scan takes
 $\mathcal{O}(1000)$ runs



Neural Boltzmann Equations

Continuous ansatz
(neural distribution functions) ✓

Monte Carlo tools from LHC ✓

Single conditional solve ✓

1) Neural distribution functions

Learnable parametrization ω
inspired by Fermi-Dirac/Bose-Einstein:

$$f(y) = \frac{4\pi^2 E}{y^2} \eta p(y)$$

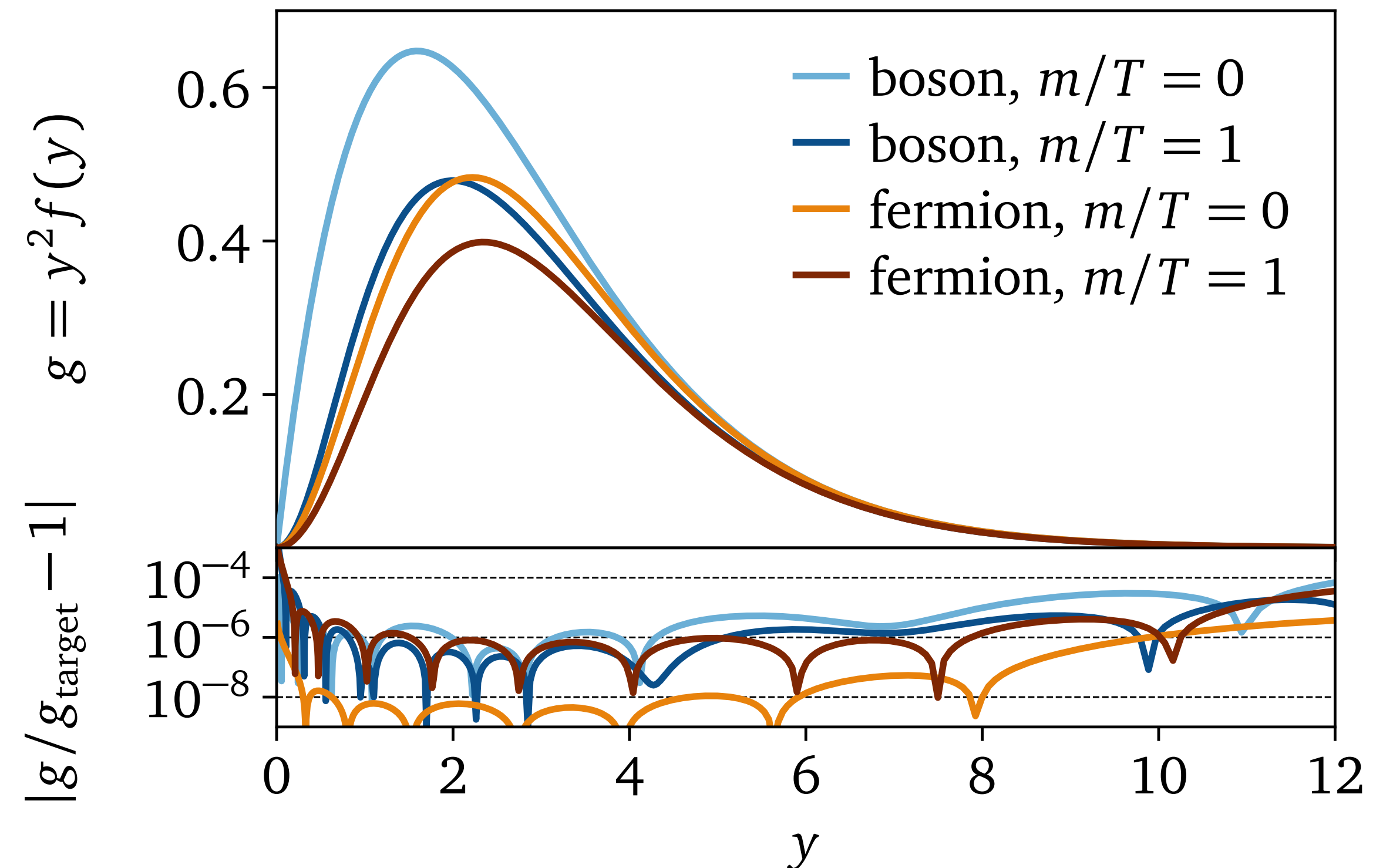
$$p(y) = \frac{y}{ET} \sum_{m=1}^M w_m \gamma(\epsilon \mid \alpha_m, \lambda_m)$$

$$\epsilon = \frac{\sqrt{z^2 + m^2} - m}{T}$$

$$\gamma(\epsilon \mid \alpha, \lambda) = \frac{\lambda}{\Gamma(\alpha)} (\lambda \epsilon)^{\alpha-1} e^{-\lambda \epsilon}$$

Predict parametrisation based on theory ξ :

$$\omega = \left\{ \eta, (w_m, \alpha_m, \lambda_m)_{m=1}^M \right\} = \text{NN}(\xi; \theta)$$



2) Monte Carlo collision operator

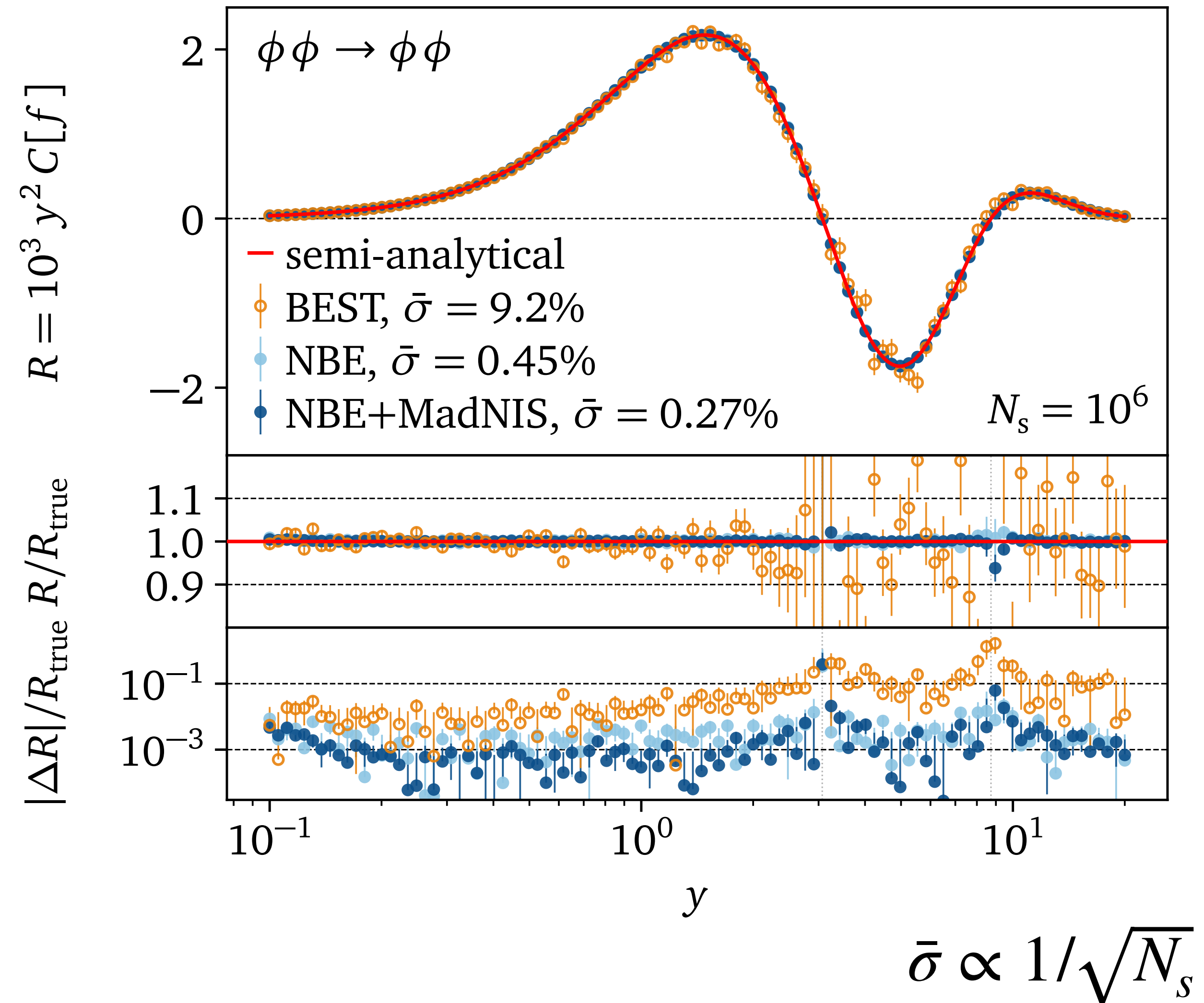
$$\mathcal{C}_a[f] = \frac{1}{2E_a} \sum_k \int d\Phi_k |\mathcal{M}_k|^2 \Lambda_k[f]$$

$$\Lambda_k[f] = \prod_{a \in k_{\text{out}}} f_a \prod_{a \in k_{\text{in}}} (1 \mp f_a) - (\text{in} \leftrightarrow \text{out})$$

Tricks (borrowed from LHC):

MadSpace mappings, $f(y)$ sampling,
MadNIS normalising flow refinement

1000 $\times$ more sample efficient
than BEST!



3) Natural gradient training

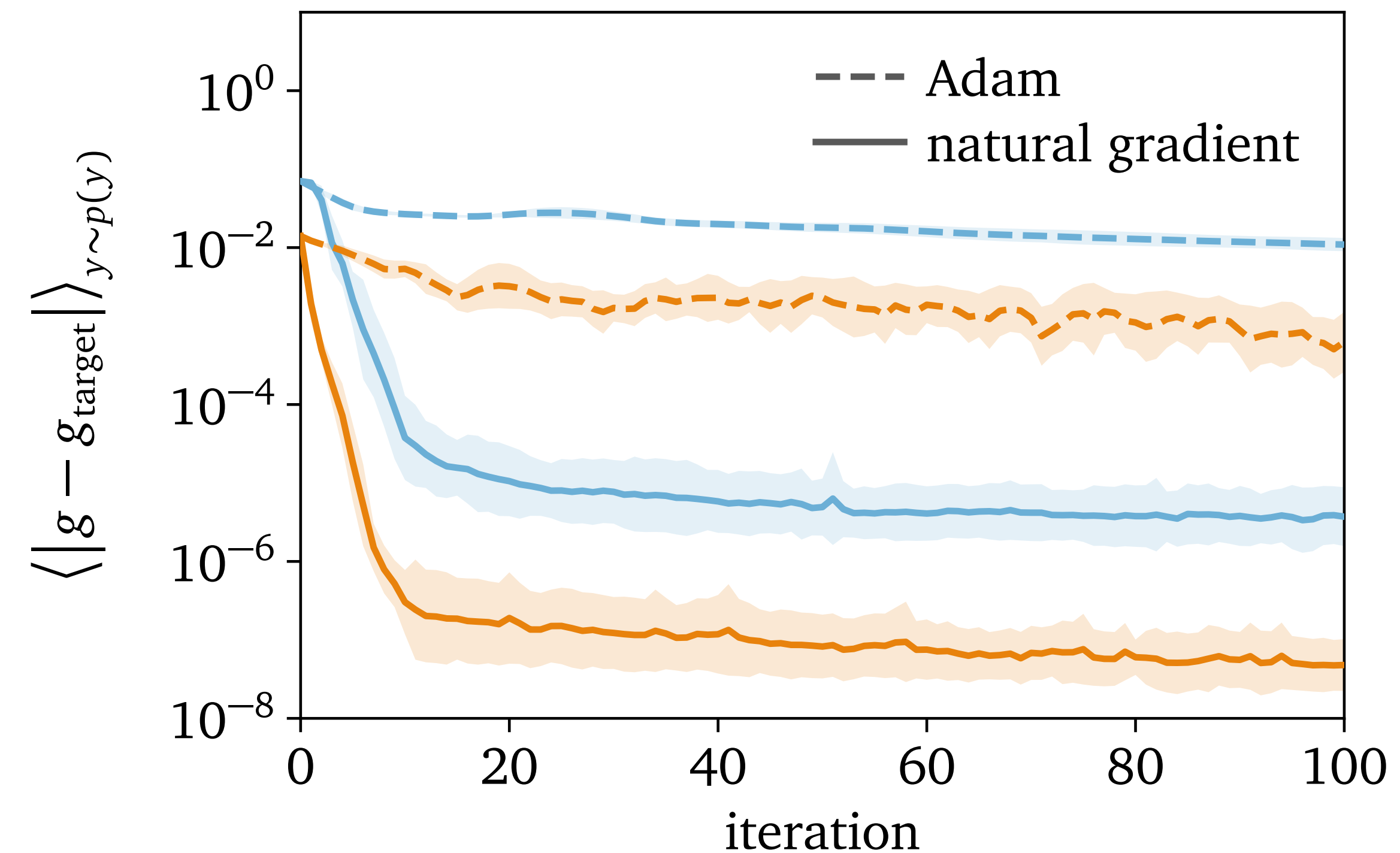
1) Build fixed-point target $g_{\text{target}}(y)$, construct residual $r = g(y; \theta) - g_{\text{target}}(y)$, weights w

2) Natural gradient step

$$J = \frac{\partial g(y; \theta)}{\partial \theta} \quad W = \text{diag}(w) \quad F = J^\top W J$$

$$\theta' = \theta - F^{-1} J^\top W r$$

Alternatively: $\theta' = \theta - J^\top (J J^\top)^{-1} r$



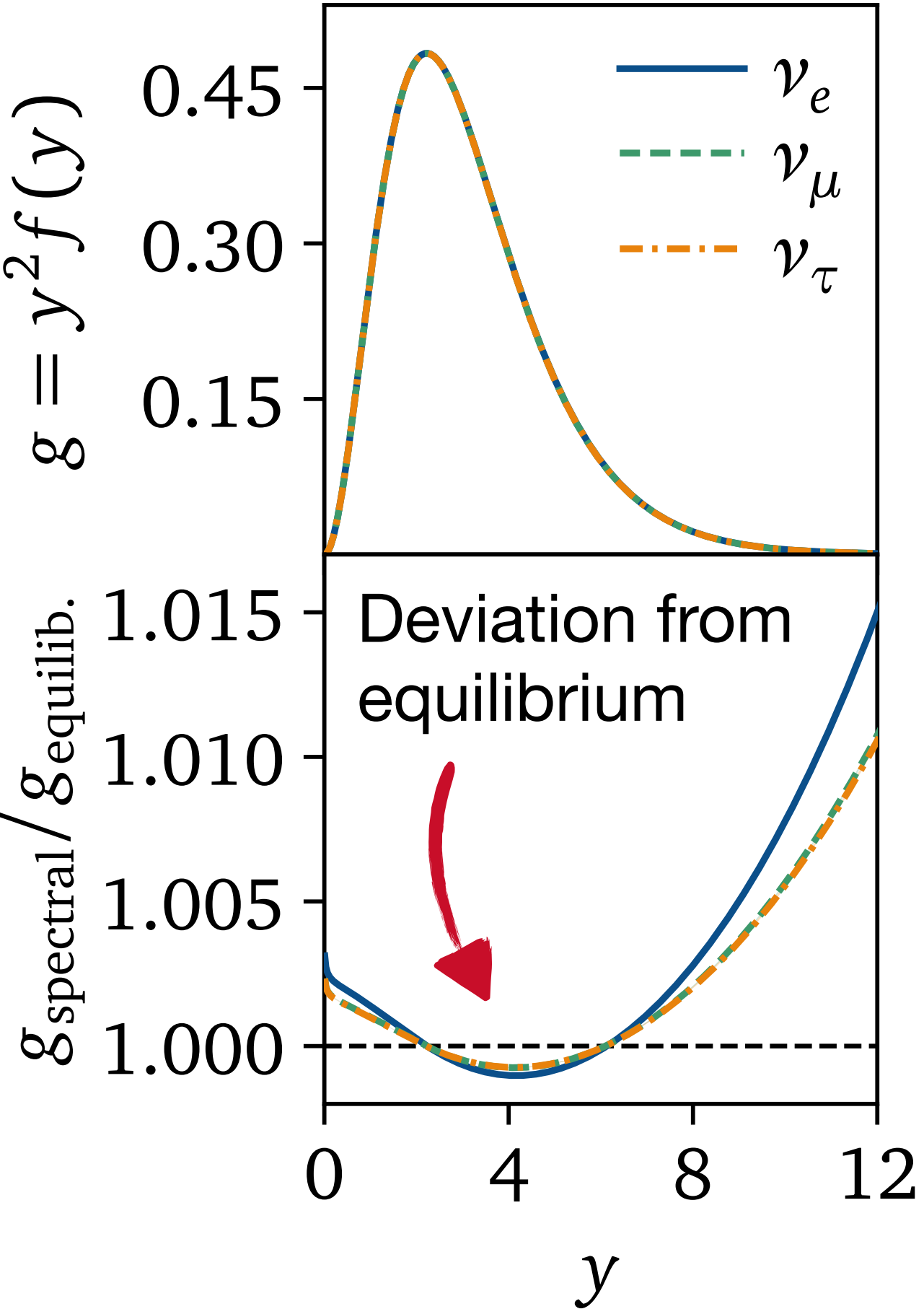
Neutrino decoupling

Early universe precision observable:

$$N_{\text{eff}} = 3.0440 \pm 0.0001$$

NBE matches precision of literature value

Description	NBE	Literature
Instantaneous decoupling	—	3 (exact)
Non-instantaneous	3.0340 ± 0.0001	3.0350 ± 0.0010 (Escudero Abenza, 2020)
+ finite-temperature QED	3.0434 ± 0.0001	$3.0443 \pm 0.0010^*$ (Escudero Abenza, 2020)
+ spectral distortions	3.0434 ± 0.0001	$3.0434 \pm 0.0001^*$ (Bennett et al., 2021)
+ flavour oscillations	3.0439 ± 0.0001	3.0440 ± 0.0001 (Bennett et al., 2021)



Neutrino decoupling with new particles

Theory (SM): $N_{\text{eff}} = 3.0440 \pm 0.0002$



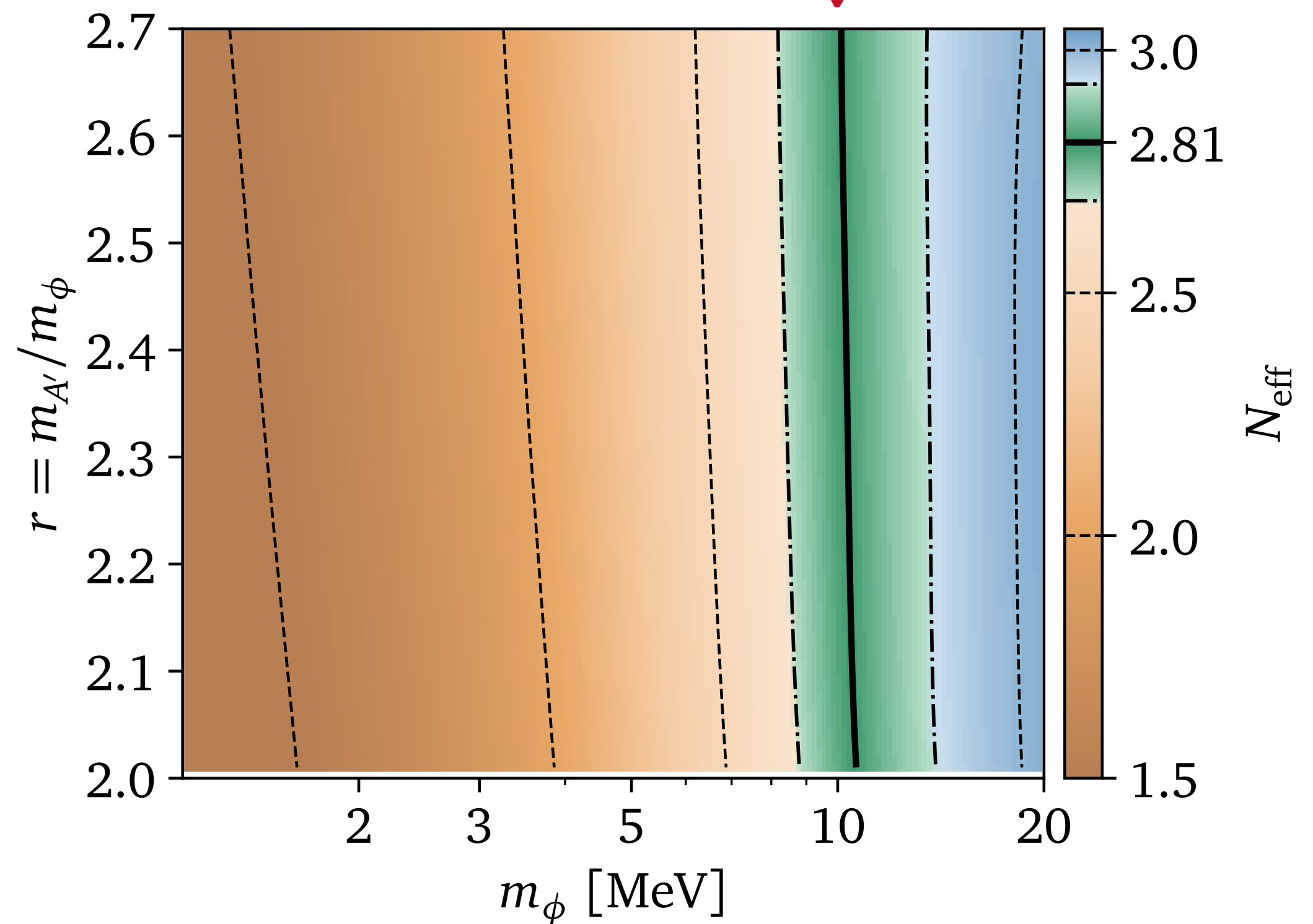
Experiment: $N_{\text{eff}} = 2.81 \pm 0.12$



agrees

Extended theory with
new particles

single NBE training



Summary

- NBE can do fast+precise cosmic evolution calculations
- Use LHC event generation tools for cosmology
- The natural gradient method is cool!

Learn more
at poster #65



Neural Boltzmann Equations

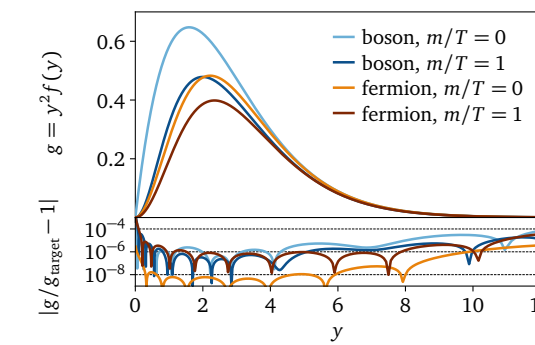
J. Spinner¹, J. D. Shergold²

¹Institute for Particle Physics Phenomenology, Durham University, United Kingdom
²Department of Mathematical Sciences, University of Liverpool, United Kingdom



Neural distribution functions

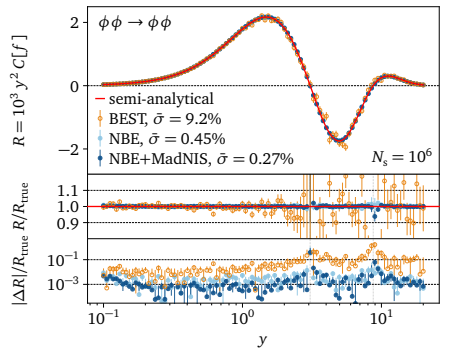
- Factorise $f(y)$ into normalisation η and normalised shape $p(y)$
$$f(y) = \frac{4\pi^2 E}{y^2} \eta p(y), \quad \eta = \int dy \frac{y^2}{4\pi^2 E} f(y), \quad \int dy p(y) = 1$$
- Density $p(y)$ as gamma mixture in the kinetic energy ε
$$p(y) = \frac{y}{ET} \sum_{m=1}^M w_m \gamma(\varepsilon | \alpha_m, \lambda_m), \quad \gamma(\varepsilon | \alpha, \lambda) = \frac{\lambda}{\Gamma(\alpha)} (\lambda \varepsilon)^{\alpha-1} e^{-\lambda \varepsilon}, \quad \varepsilon = \frac{\sqrt{y^2 + m^2} - m}{T}$$
- Ensure $f \leq 1$ for fermions with an unconstrained f_0 as $f = f_0 / (1 + f_0)$
- Predict parameters ω with NN conditioned on theory ξ
$$\omega = \{\eta, (w_m, \alpha_m, \lambda_m)_{m=1}^M\} = \text{NN}(\xi; \theta)$$



Collision operator

- MC integral over all other particles
$$C_a[f] = \frac{1}{2E_a} \sum_k \int d\Phi_k |\mathcal{M}_k|^2 \Lambda_k[f]$$

$$\Lambda_k[f] = \prod_{b \in k_{\text{out}}} f_b \prod_{c \in k_{\text{in}}} (1 \pm f_c) \quad (\text{in} \leftrightarrow \text{out})$$
- MadSpace [2] mappings, $f(y)$ sampling, MadNIS flow refinement
- NBE 1000× more sample efficient than BEST baseline [3]



Algorithm 1 Phase space generation for a process with m incoming and $n - m$ outgoing particles.
Require: observed momentum y , condition ξ , masses $m_1, \dots, m_n$, proposal densities $p_1, \dots, p_n$
1: $y_i \leftarrow y_i \xi$
2: $y_k \sim p_k(\cdot | \xi)$, $\hat{\Omega}_k \sim \mathcal{U}(S^2)$ for $k = 2, \dots, m$
3: $y_{m+1} \sim \mathcal{U}([0, 1]^{n-m-1})$
4: If neural importance sampling then
5: $u_i, l_{\text{out}} \leftarrow \text{ToUnitCube}(y_1, \dots, y_m, \hat{\Omega}_1, \dots, \hat{\Omega}_m, u_{\text{out}})$
6: $u_i, l_{\text{out}} \leftarrow \text{Flow}(u | y_i, \xi)$
7: $(y_1, \dots, y_m, \hat{\Omega}_1, \dots, \hat{\Omega}_m, u_{\text{out}}), l_{\text{out}} \leftarrow \text{ToUnitCube}^{-1}(u)$
8: else
9: $l_{\text{out}}, l_{\text{out}} \leftarrow 1$
10: $\hat{y}_i \sim y_i \hat{\Omega}_i$ for $k = 2, \dots, m$
11: $\sqrt{s} = ((\sum_{i=1}^m E_i)^2 - |\sum_{i=1}^m \hat{y}_i|^2)^{1/2}$ with $E_i = (y_i^2 + m_i^2)^{1/2}$
12: $\hat{y}_{m+1}, \dots, \hat{y}_n, l_{\text{out}} \leftarrow \text{MadSpace}(u_{\text{out}}; \sqrt{s}, m_{m+1}, \dots, m_n)$
13: $y_{m+1}, \dots, y_n, l_{\text{out}} \leftarrow \text{Flow}(y_{m+1}, \dots, y_n, l_{\text{out}})$
14: **return** $(y_1, \dots, y_n, l_{\text{out}})$

Boltzmann equations

- Solve for particle distribution function $f_a(y)$ evolution in expanding universe
- Comoving quantities: time x , momentum y
- Examples: dark matter production, baryogenesis/leptogenesis, neutrino decoupling

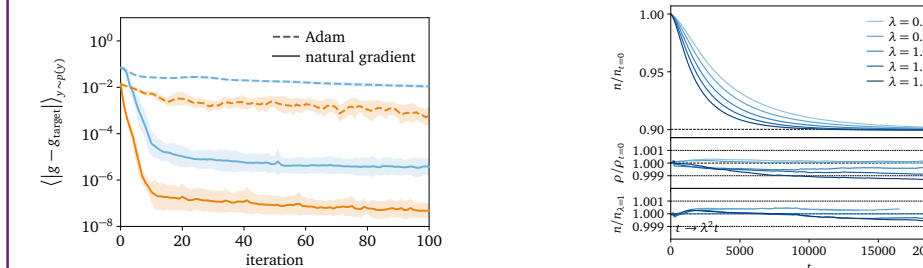
$$\frac{\partial f_a(y)}{\partial \log x} = \frac{1}{H} C_a[f]$$

Challenges

- Increase overall numerical precision
- Advanced collision integrals (higher-multiplicity processes, NLO)
- Flexible and easy to use framework
- Theory parameter conditioned training for efficient scans

Boltzmann equation solver

- Fixed-point iteration with stiffness-aware target
- Natural gradient updates include function space geometry



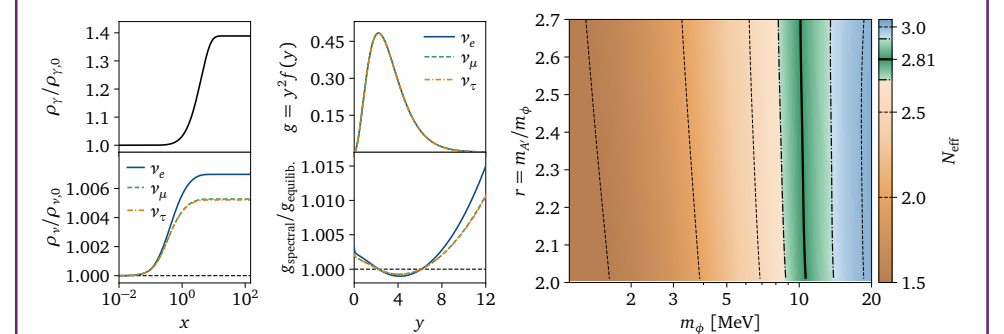
Algorithm 2 Boltzmann solver step $\log x_n \rightarrow \log x_{n+1}$ fixed-point iteration
Require: condition prior $p_{\text{prior}}(\xi)$, frozen previous solutions $\theta_1, \theta_2, \dots$
1: $\theta_{n+1} \leftarrow \theta_n$
2: **repeat**
3: $\xi_i \sim p_{\text{prior}}(\xi)$ for $i = 1, \dots, B$
4: $\xi \leftarrow (\xi_1, \dots, \xi_B)$
5: $\omega_i \leftarrow \text{NN}(\xi; \theta)$ for $i = 1, \dots, N$
6: **for** species $a = 1, \dots, N$ **do**
7: $y \sim p(y | \omega_{a+1}, \omega_a)$
8: $s(y), \kappa(y) \leftarrow$ evaluate collision integral
9: $A(y), \gamma \leftarrow$ evaluate scheme coefficients
10: $\hat{g}_{\text{approx}}(y) \leftarrow T(g(y; \omega_{a+1}, \omega_a), s(y), A(y), \gamma)$
11: $r_i \leftarrow g(y; \omega_{a+1}, \omega_a) - \hat{g}_{\text{approx}}(y)$
12: $w \leftarrow [r(y; \omega_{a+1}, \omega_a) g(y; \omega_{a+1}, \omega_a)]^{-1}$
13: $J \leftarrow \frac{\partial \hat{g}_{\text{approx}}}{\partial \omega}$ using autodiff
14: $W \leftarrow \text{diag}(w), F \leftarrow J^T W J$
15: $\theta_{n+1,2} \leftarrow \theta_{n+1,2} - F^{-1} J^T W r$
16: **until** convergence criteria satisfied

Example: neutrino decoupling

- Precision Standard Model calculation of N_{eff} , the effective number of relativistic neutrino degrees of freedom
- Excellent agreement with the literature result $N_{\text{eff}} = 3.0440 \pm 0.0001$ [4]

Description	NBE	Literature
Instantaneous decoupling	—	3 (exact)
Non-instantaneous	3.0340 ± 0.0001	3.0350 ± 0.0010
+ finite-temperature QED	3.0434 ± 0.0001	3.0443 ± 0.0010
+ spectral distortions	3.0434 ± 0.0001	3.0434 ± 0.0001
+ flavour oscillations	3.0439 ± 0.0001	3.0440 ± 0.0001

- Conditioned NBE scans the parameter space of a BSM model [5] that can resolve the 2σ tension with $N_{\text{eff}} = 2.81 \pm 0.12$ [6]



References

- [1] Spinner, Shergold, "Neural Boltzmann Equations." *arXiv:2608*. (2026).
- [2] Heimel, Mattelaer, Winterhalder, "MadSpace – Event Generation for the Era of GPUs and ML." *arXiv:2602.06895* (2026).
- [3] Yoon, "Boltzmann equation solver for thermalization (BEST)." *Comput. Phys. Commun.* 327, 110295 (2026), *arXiv:2603.28848*.
- [4] Froustey, Pitrou, Volpe, "Neutrino decoupling including flavour oscillations and primordial nucleosynthesis." *JCAP* 12, 015 (2020), *arXiv:2008.01074*.
- [5] Escudero, Ovchinnikov, Weiner, "What does it take to have $N_{\text{eff}} < 3$ at CMB times?" *arXiv:2603.22391* (2026).
- [6] Louis et al. (ACT), "DR6 power spectra, likelihoods and Λ CDM parameters." *JCAP* 11, 062 (2025), *arXiv:2503.14452*.

Extra slides

Algorithm 1 Phase-space generation for a process with m incoming and $n - m$ outgoing particles.

Require: observed momentum y_1 , condition ξ , masses $m_1, \dots, m_n$, proposal densities $p_2, \dots, p_m$

- 1: $\vec{y}_1 \leftarrow y_1 \hat{z}$ $\triangleright$ observed leg, pinned to a fixed axis $\hat{z}$
 - 2: $y_k \sim p_k(\cdot | \xi)$, $\hat{\Omega}_k \sim \mathcal{U}(S^2)$ for $k = 2, \dots, m$ $\triangleright$ sample initial legs
 - 3: $u_{\text{out}} \sim \mathcal{U}([0, 1]^{3(n-m)-4})$
 - 4: **if** neural importance sampling **then**
 - 5: $u, J_{\text{enc}} \leftarrow \text{ToUnitCube}(y_2, \dots, y_m, \hat{\Omega}_2, \dots, \hat{\Omega}_m, u_{\text{out}})$
 - 6: $u, J_{\text{flow}} \leftarrow \text{Flow}(u | y_1, \xi)$ $\triangleright$ MadNIS refinement
 - 7: $(y_2, \dots, y_m, \hat{\Omega}_2, \dots, \hat{\Omega}_m, u_{\text{out}}), J_{\text{dec}} \leftarrow \text{ToUnitCube}^{-1}(u)$
 - 8: **else**
 - 9: $J_{\text{enc}}, J_{\text{flow}}, J_{\text{dec}} \leftarrow 1$
 - 10: $\vec{y}_k \leftarrow y_k \hat{\Omega}_k$ for $k = 2, \dots, m$ $\triangleright$ build initial state
 - 11: $\sqrt{s} \leftarrow [(\sum_{k \leq m} E_k)^2 - |\sum_{k \leq m} \vec{y}_k|^2]^{1/2}$ with $E_k = (y_k^2 + m_k^2)^{1/2}$
 - 12: $\vec{y}_{m+1}, \dots, \vec{y}_n, J_{\text{ps}} \leftarrow \text{MadSpace}(u_{\text{out}}; \sqrt{s}, m_{m+1}, \dots, m_n)$ $\triangleright$ build final state
 - 13: $w_{\text{ps}} \leftarrow J_{\text{enc}} J_{\text{flow}} J_{\text{dec}} J_{\text{ps}} \prod_{k=2}^m \frac{\eta_k}{f_k(y_k | \xi)}$ $\triangleright$ importance weight
 - 14: **return** $(\vec{y}_1, \dots, \vec{y}_n; w_{\text{ps}})$
-

Algorithm 2 Boltzmann solver step $\log x_n \rightarrow \log x_{n+1}$ fixed-point iteration

Require: condition prior $p_{\text{prior}}(\xi)$, frozen previous solutions $\theta_n, \theta_{n-1}, \dots$

- 1: $\theta_{n+1} \leftarrow \theta_n$ $\triangleright$ warm-start from previous step
 - 2: **repeat**
 - 3: $\xi_i \sim p_{\text{prior}}(\xi)$ for $i = 1, \dots, B$ $\triangleright$ sample conditions
 - 4: $\xi \leftarrow (\xi_1, \dots, \xi_B)$ $\triangleright$ batched evaluation
 - 5: $\omega_l \leftarrow \text{NN}(\xi; \theta_l)$ for $l = n+1, n, \dots$ $\triangleright$ cache distribution function parameters
 - 6: **for** species $a = 1, \dots, N$ **do** $\triangleright$ update each species independently
 - 7: $y \sim p(\cdot | \omega_{n+1,a})$ $\triangleright$ sample test momenta
 - 8: $s(y), \kappa(y) \leftarrow$ evaluate collision integral following Section 4.2
 - 9: $A(y), \gamma \leftarrow$ evaluate scheme coefficients using Appendix A.3
 - 10: $g_{\text{target}}(y) \leftarrow T(g(y; \omega_{n+1,a}), s(y), \kappa(y), A(y), \gamma)$ $\triangleright$ fixed-point target Eq. (14)
 - 11: $r_i \leftarrow g(y; \omega_{n+1,a}) - g_{\text{target}}(y)$
 - 12: $w \leftarrow [p(y | \omega_{n+1,a}) g(y; \omega_{n+1,a})]^{-1}$
 - 13: $J \leftarrow \left. \frac{\partial g(y; \omega(\theta))}{\partial \theta} \right|_{\theta=\theta_{n+1,a}}$ using autodiff
 - 14: $W \leftarrow \text{diag}(w), F \leftarrow J^\top W J$
 - 15: $\theta_{n+1,a} \leftarrow \theta_{n+1,a} - F^{-1} J^\top W r$ $\triangleright$ natural-gradient step Eq. (7)
 - 16: **until** convergence criteria satisfied
-