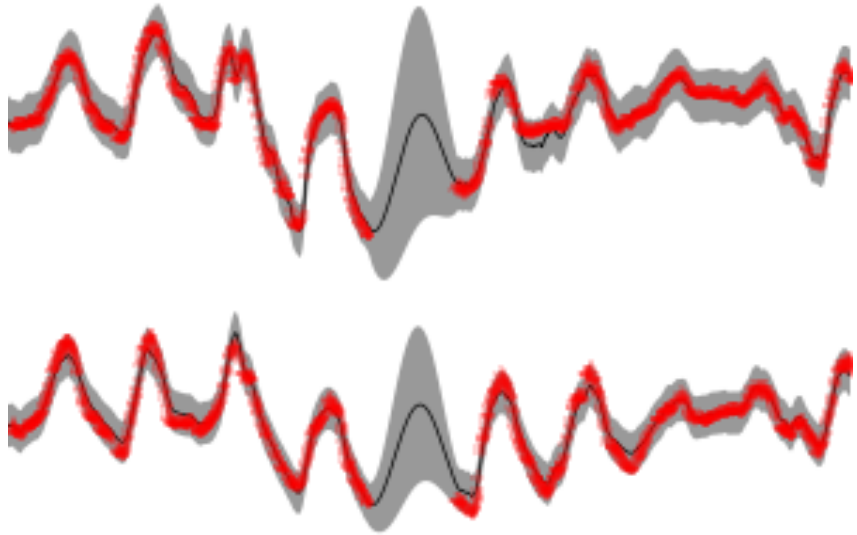
Fast Variational Bayesian Linear State-Space Model

Jaakko Luttinen

Aalto University
Department of Information and Computer Science
jaakko.luttinen@aalto.fi

September 25, 2013

Introduction



- *Model:* Gaussian linear state-space model
- *Problem:* Variational Bayesian (VB) estimation is slow
- *Solution:* Use parameter expansion

Linear state-space model

- Observations:

$$\mathbf{y}_n = \mathbf{C}\mathbf{x}_n + \text{noise}\,.$$

- Dynamics in the latent space:

$$\mathbf{x}_n = \mathbf{A}\mathbf{x}_{n-1} + \text{noise}\,.$$

- $\mathbf{x}_n$, $\mathbf{A}$ and $\mathbf{C}$ are unknown and estimated using the data.

Variational Bayesian learning

- The variables are given Gaussian priors (hyperparameters are ignored for simplicity)
- Approximate the posterior as

$$q(\mathbf{x}_1, \ldots, \mathbf{x}_N) \cdot q(\mathbf{A}) \cdot q(\mathbf{C})$$

- Update the variables one at a time

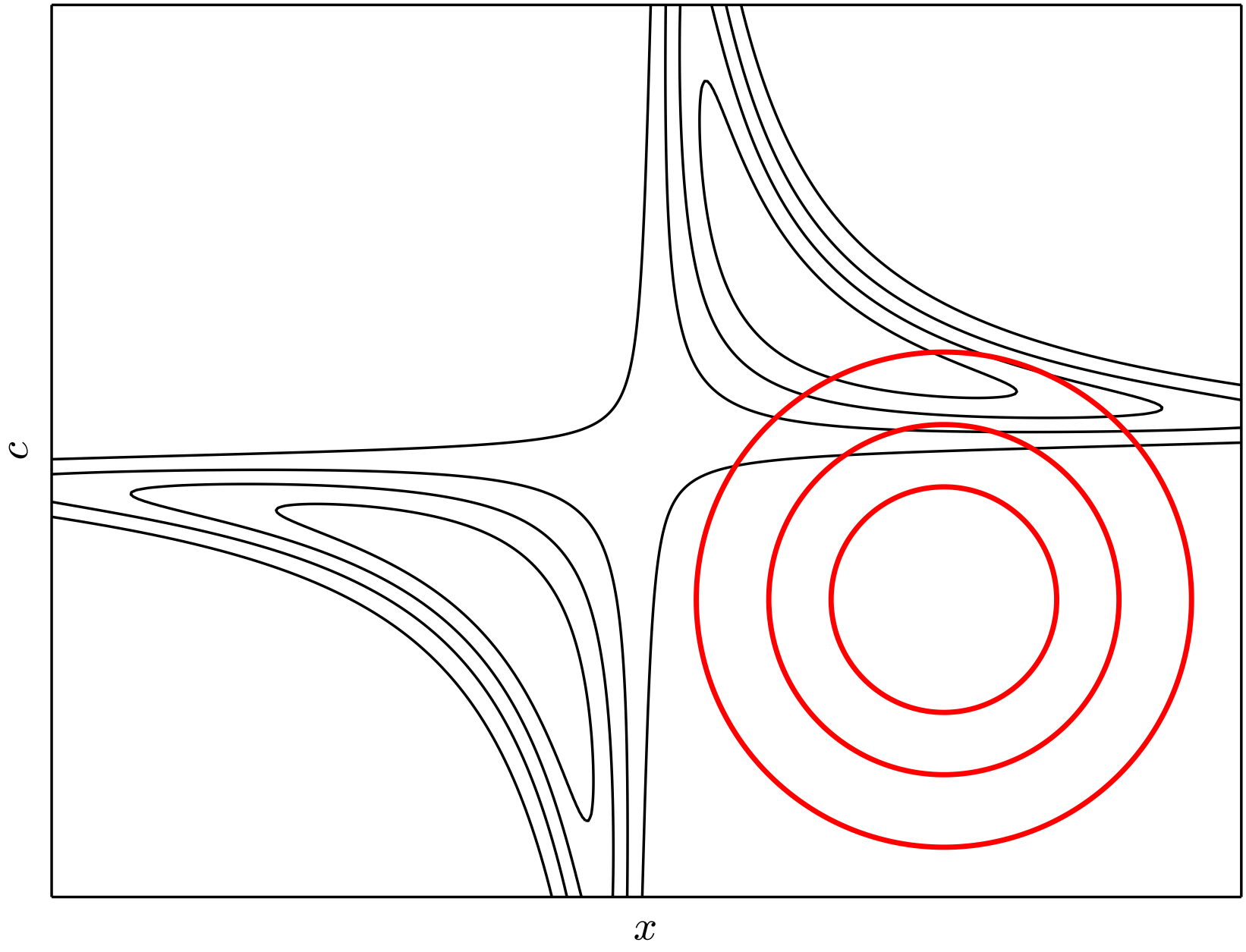
Simple demonstration

- 1-dimensional model $y = cx + \text{noise}$

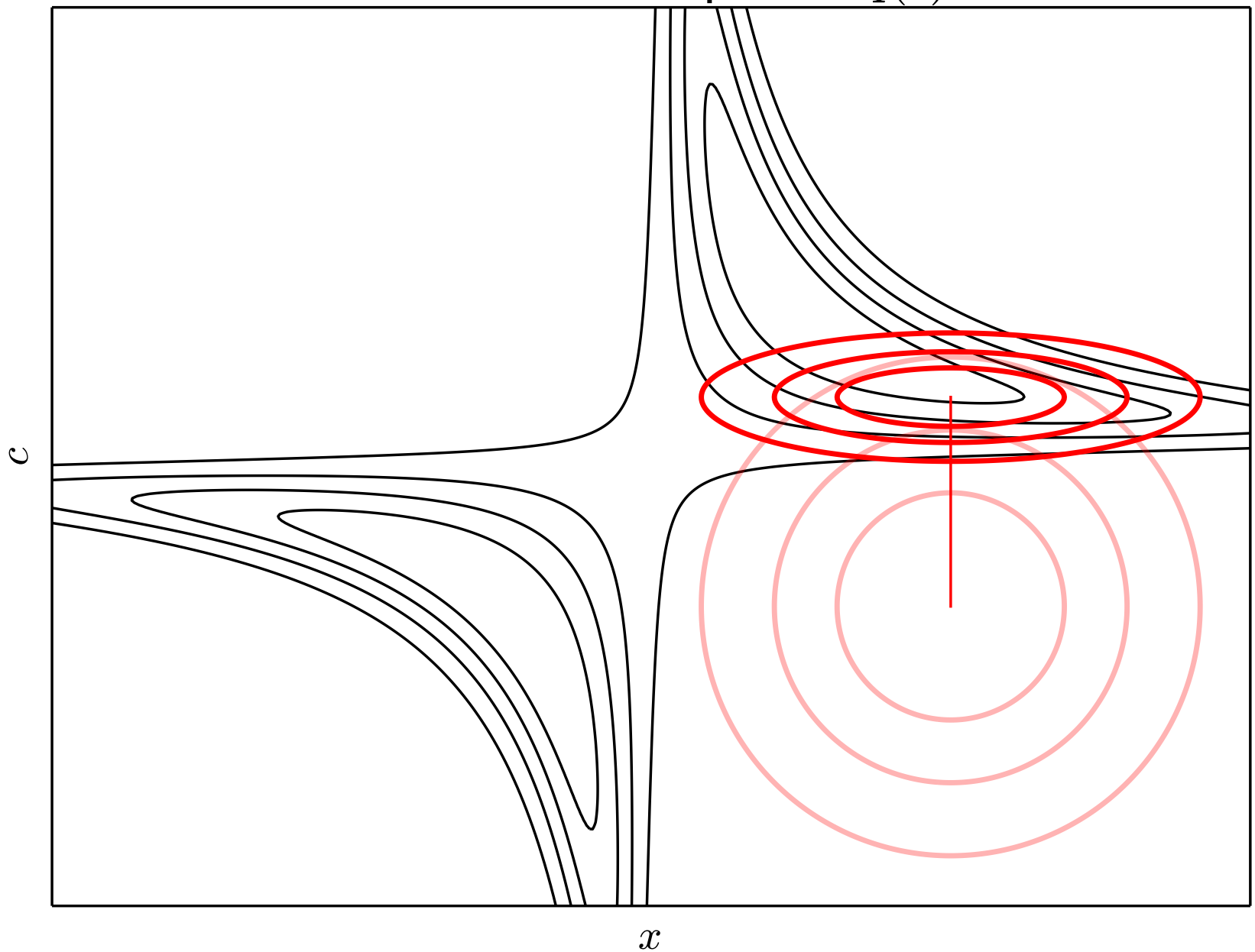
- Approximate VB posterior:

$$p(c, x|y) \approx q(c) \cdot q(x)$$

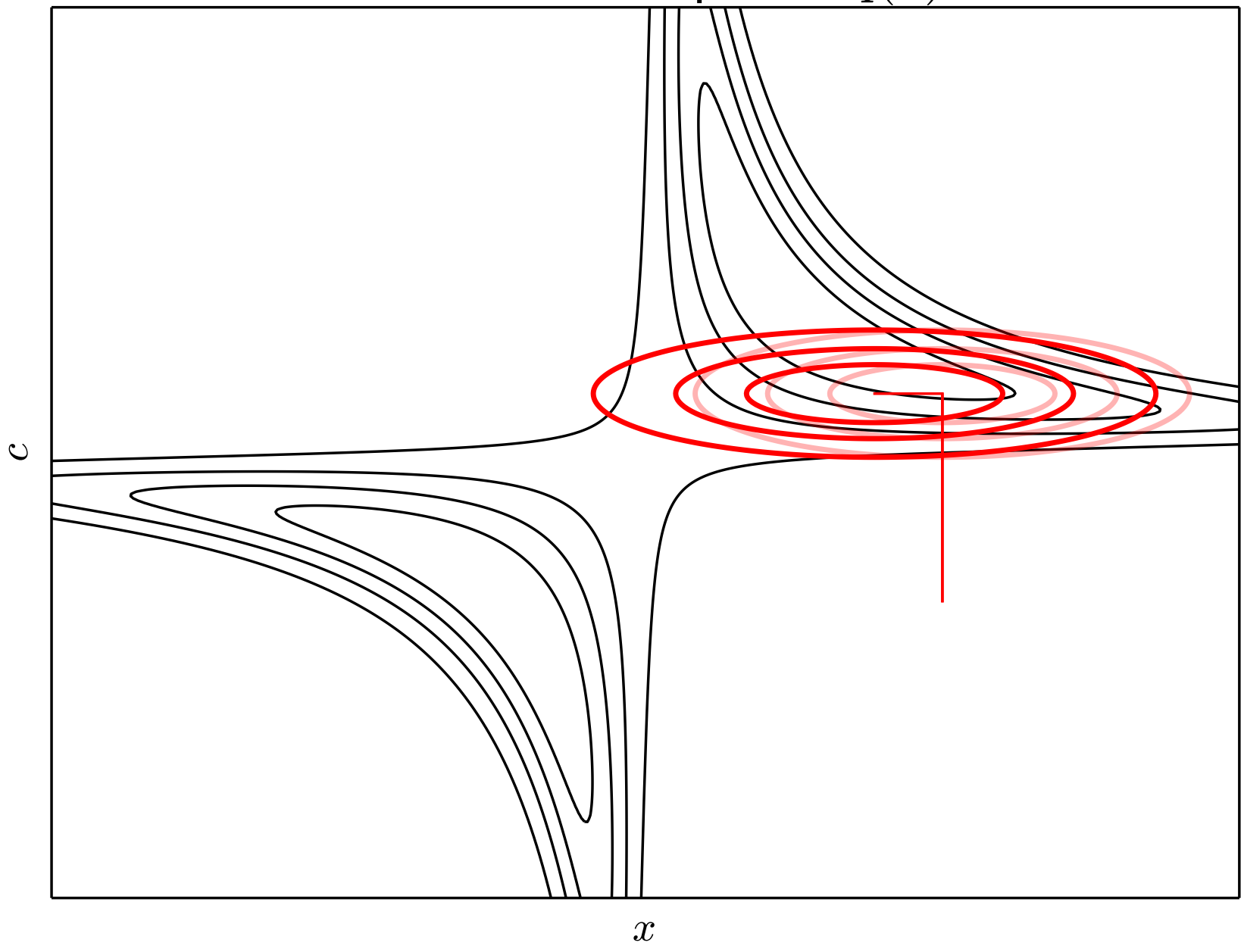
Initialization



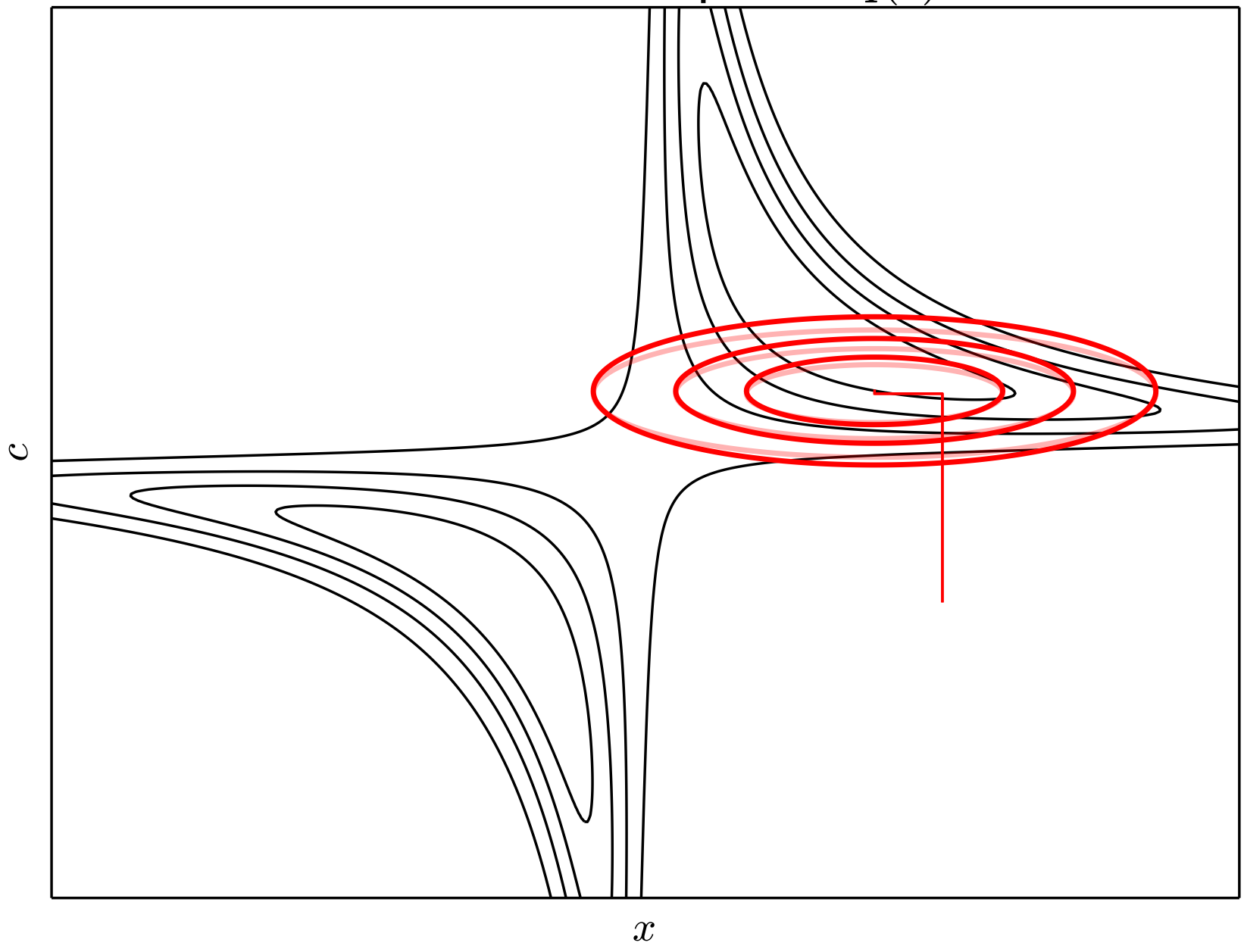
Iteration 1 - Update $q(c)$



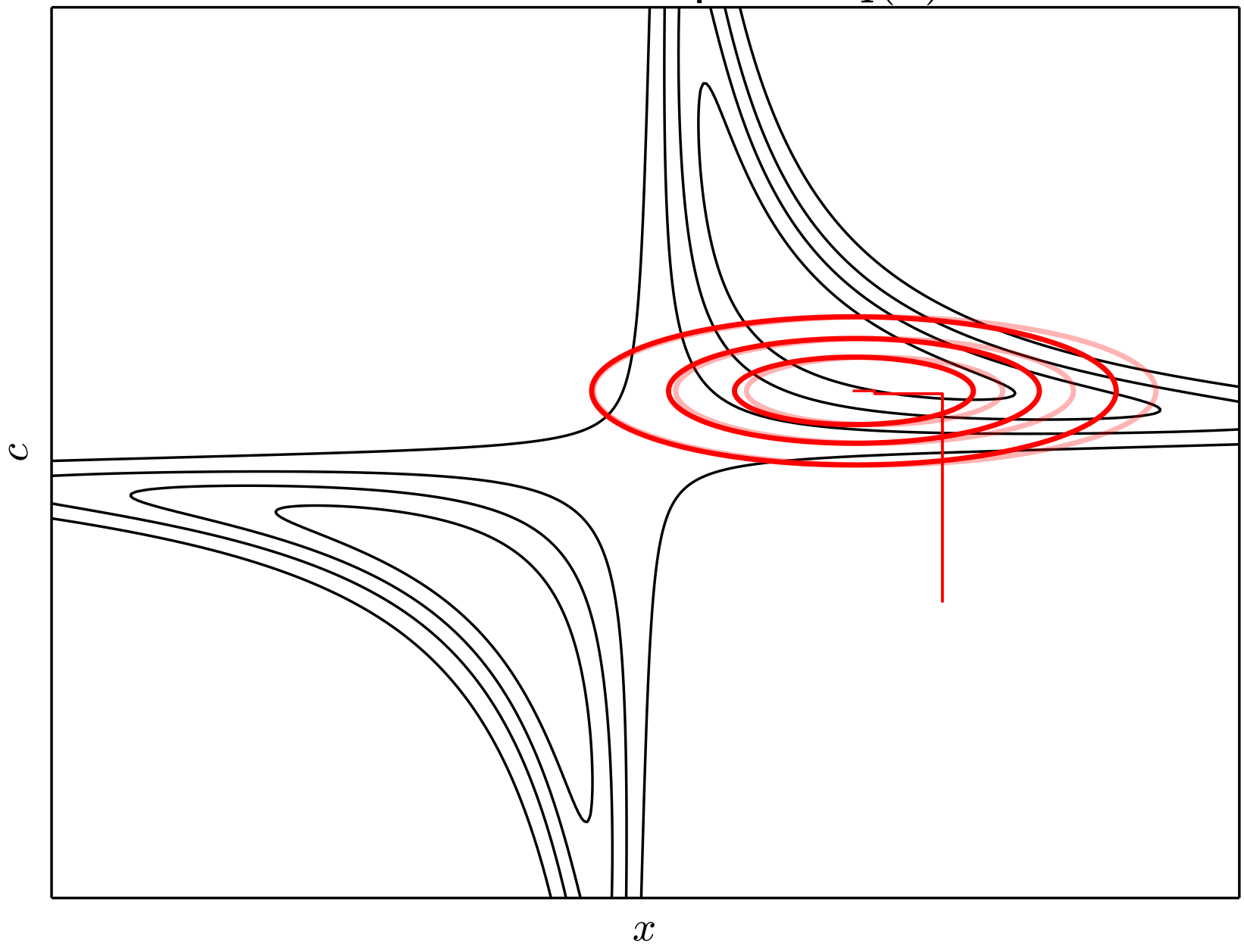
Iteration 1 - Update $q(x)$



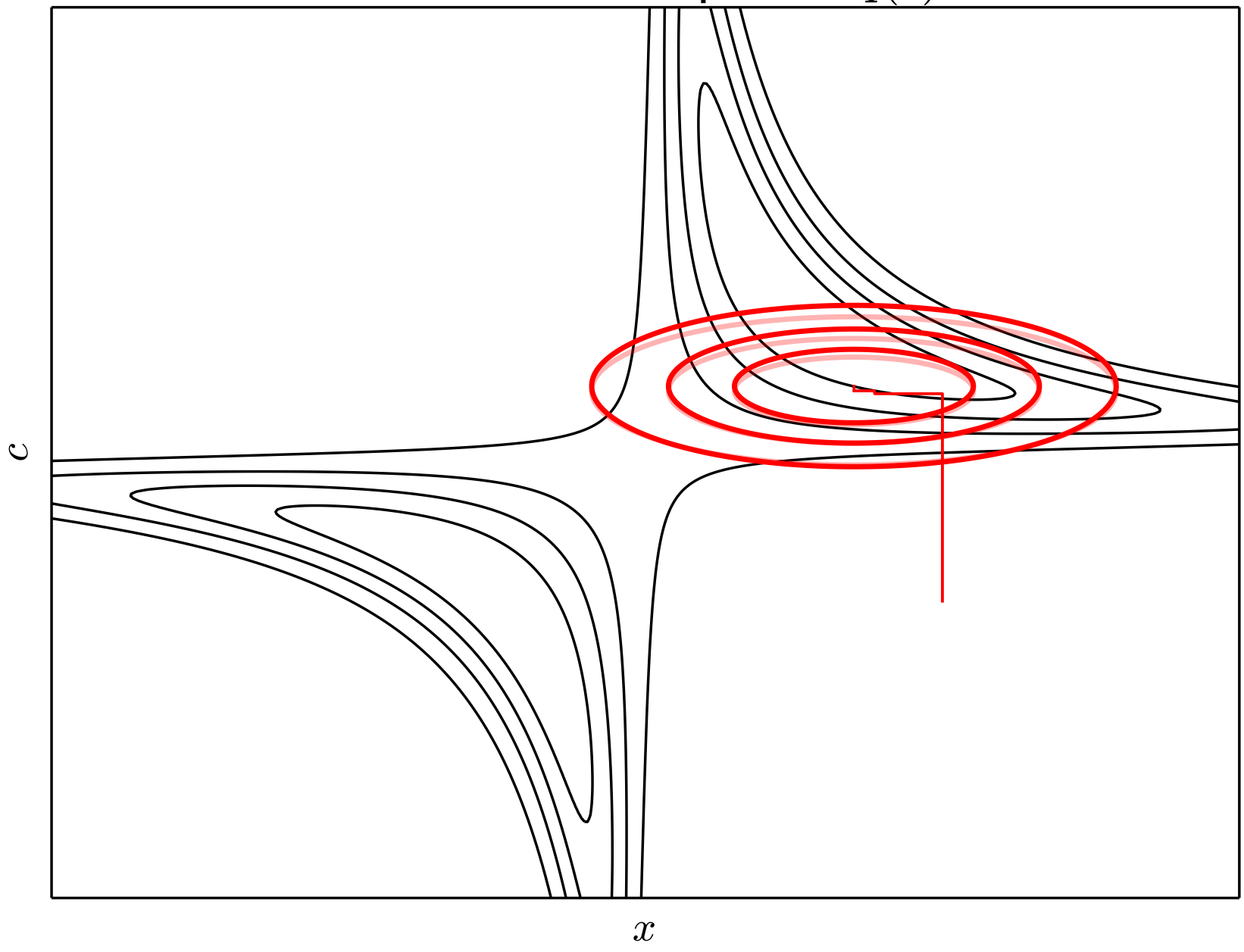
Iteration 2 - Update $q(c)$



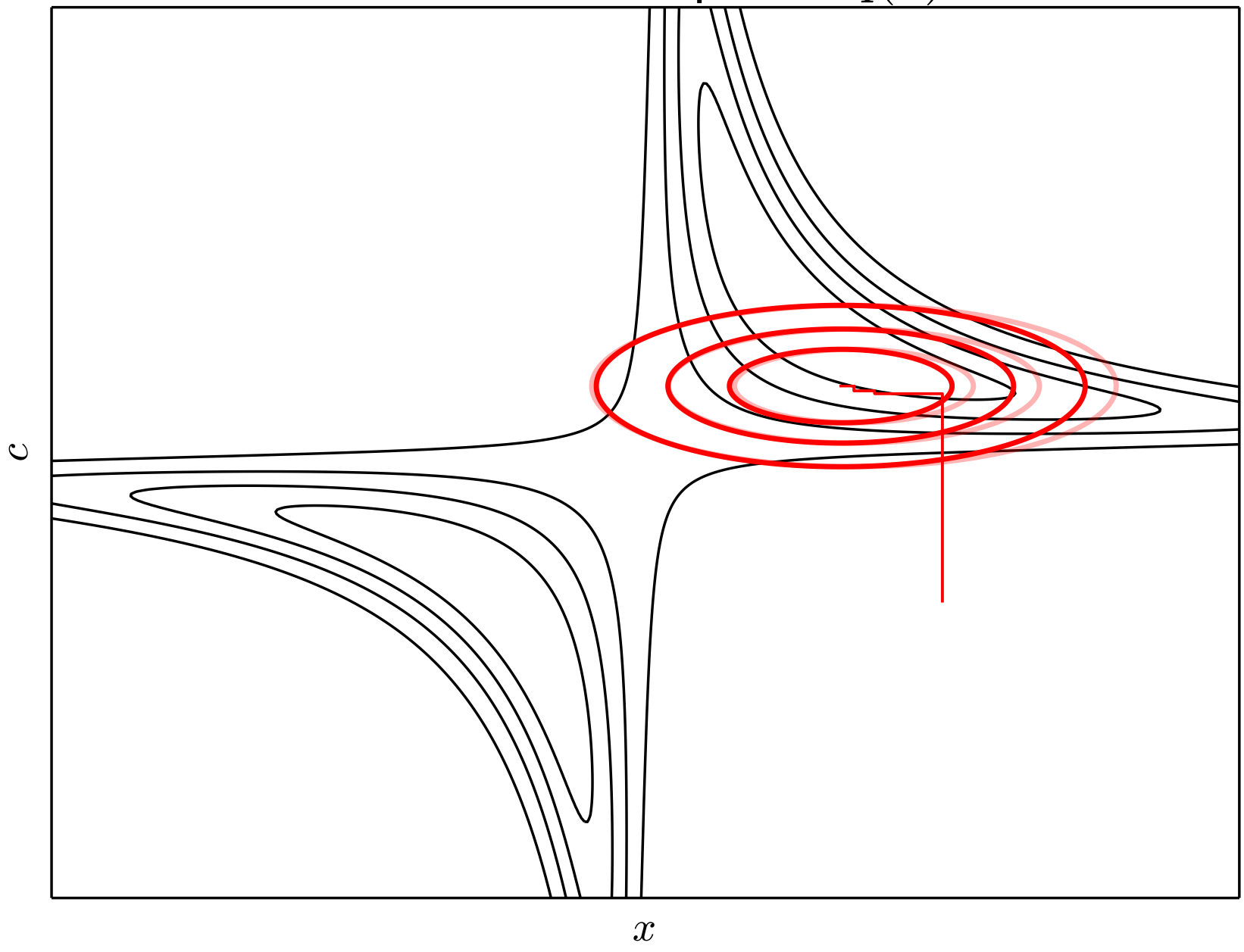
Iteration 2 - Update $q(x)$



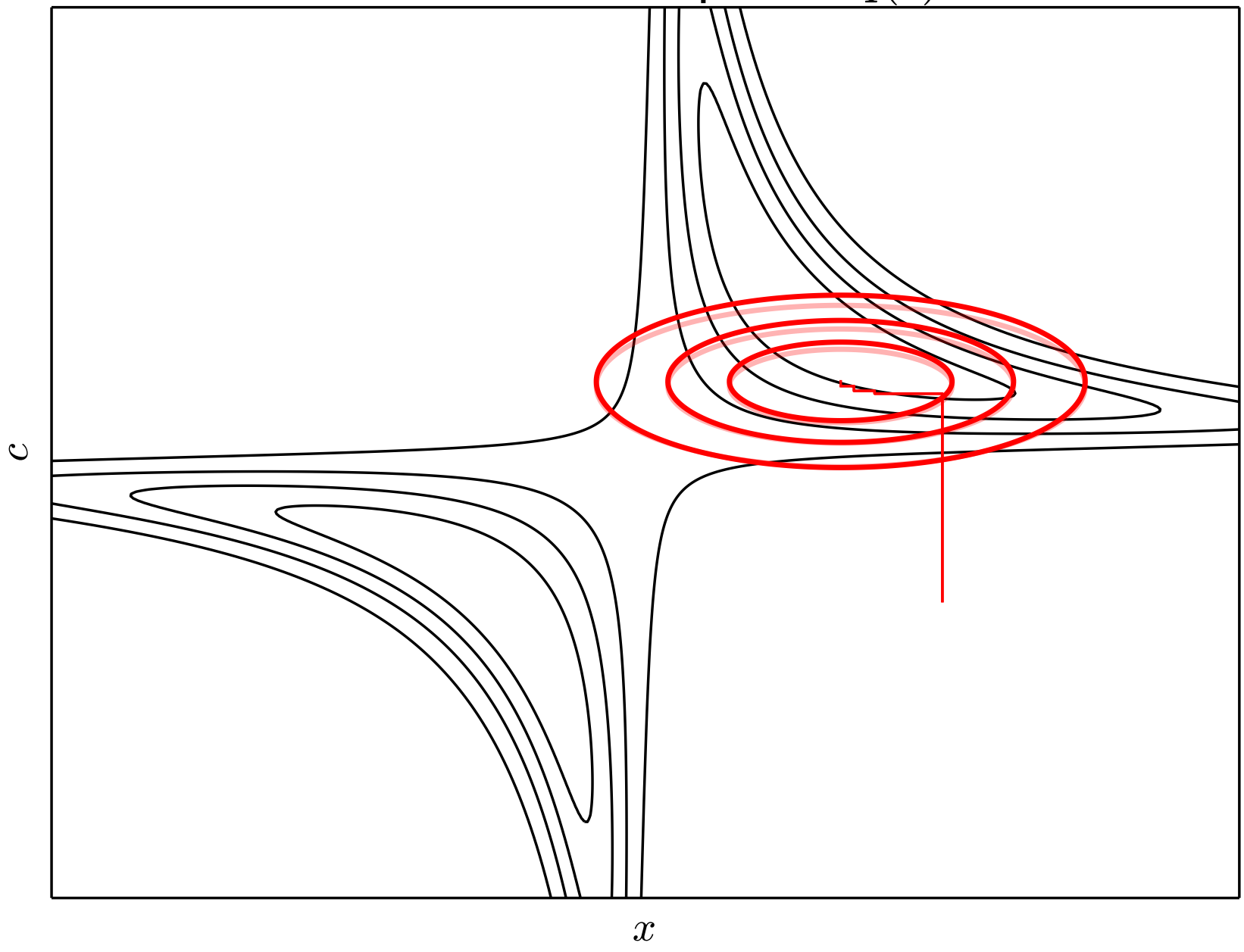
Iteration 3 - Update $q(c)$



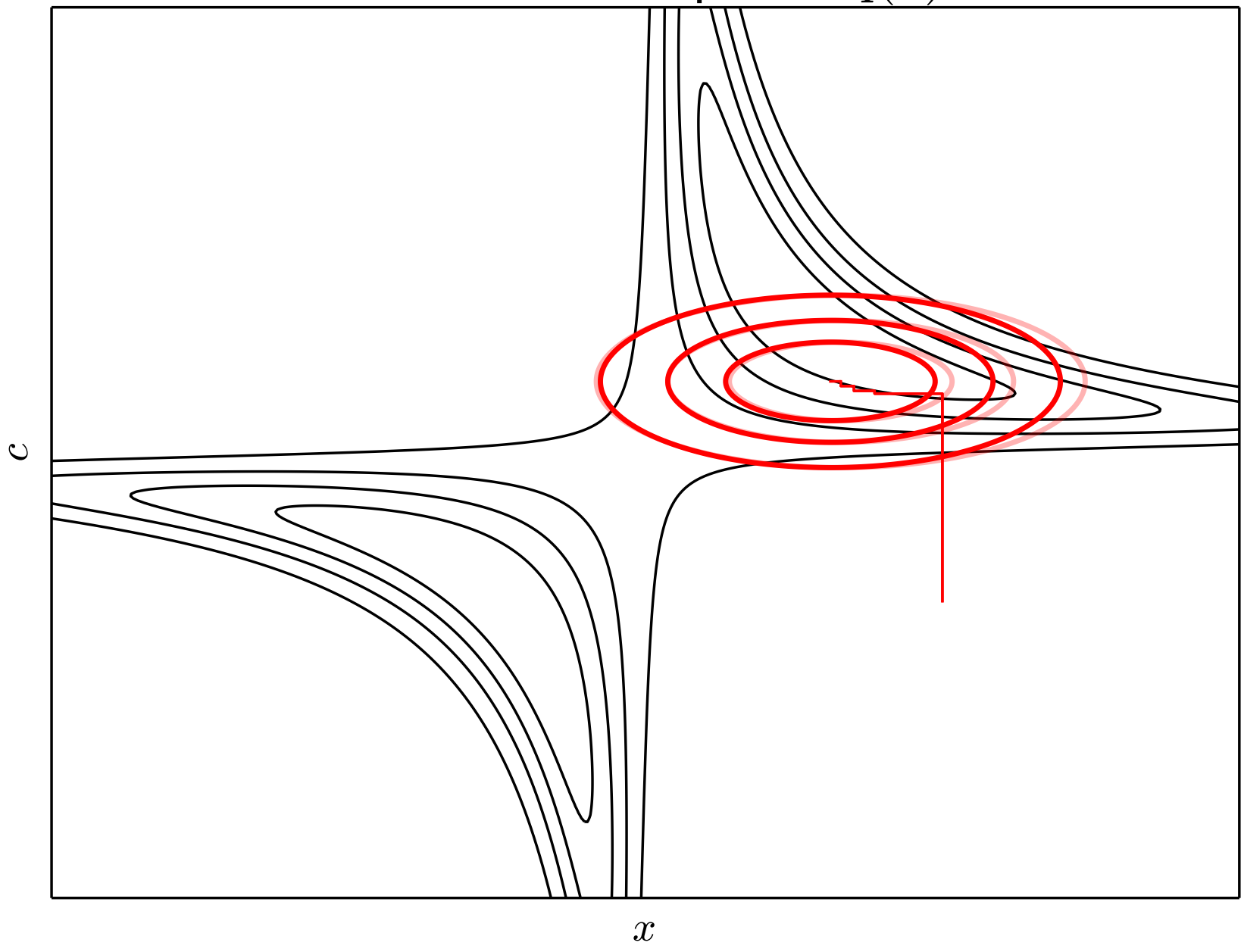
Iteration 3 - Update $q(x)$



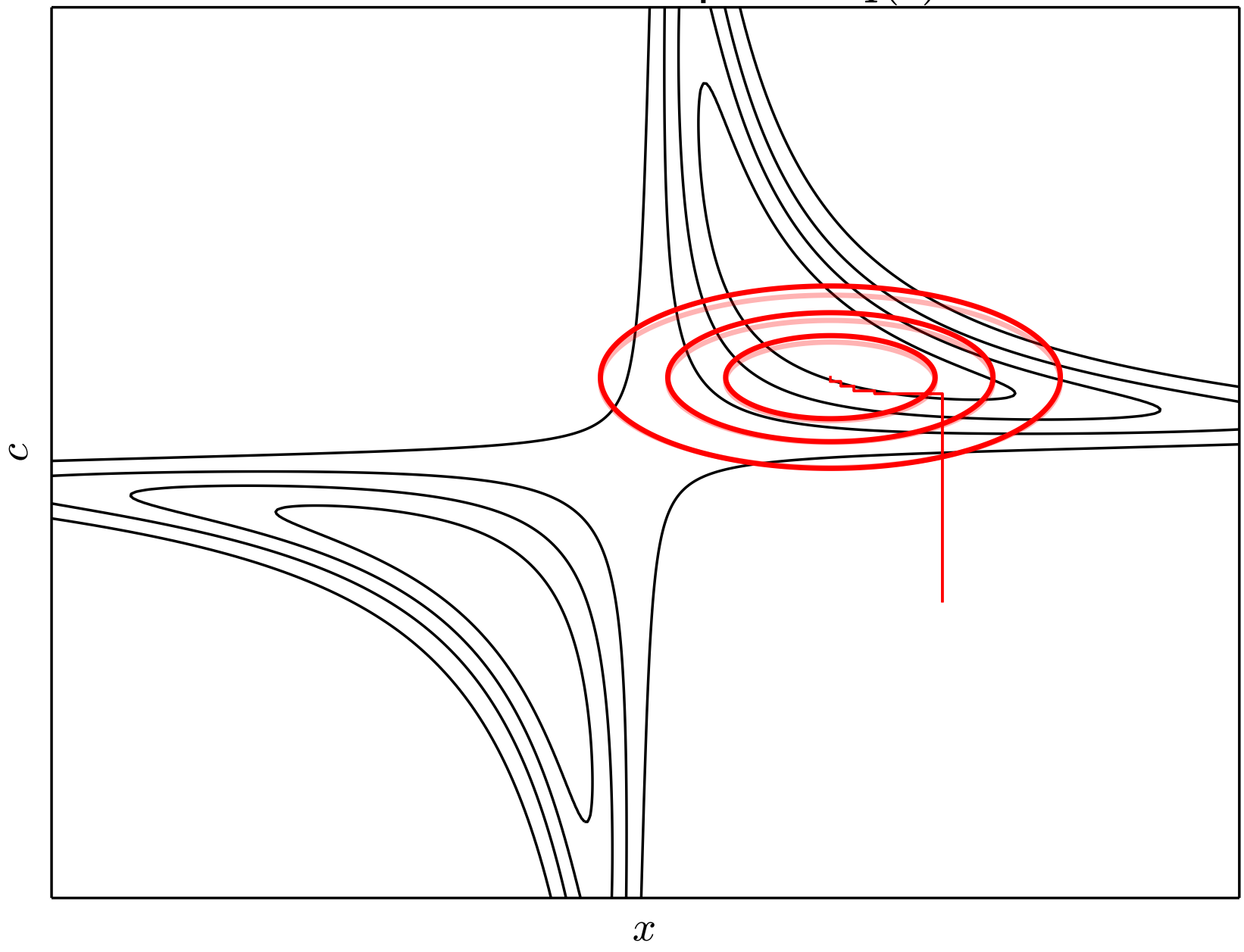
Iteration 4 - Update $q(c)$



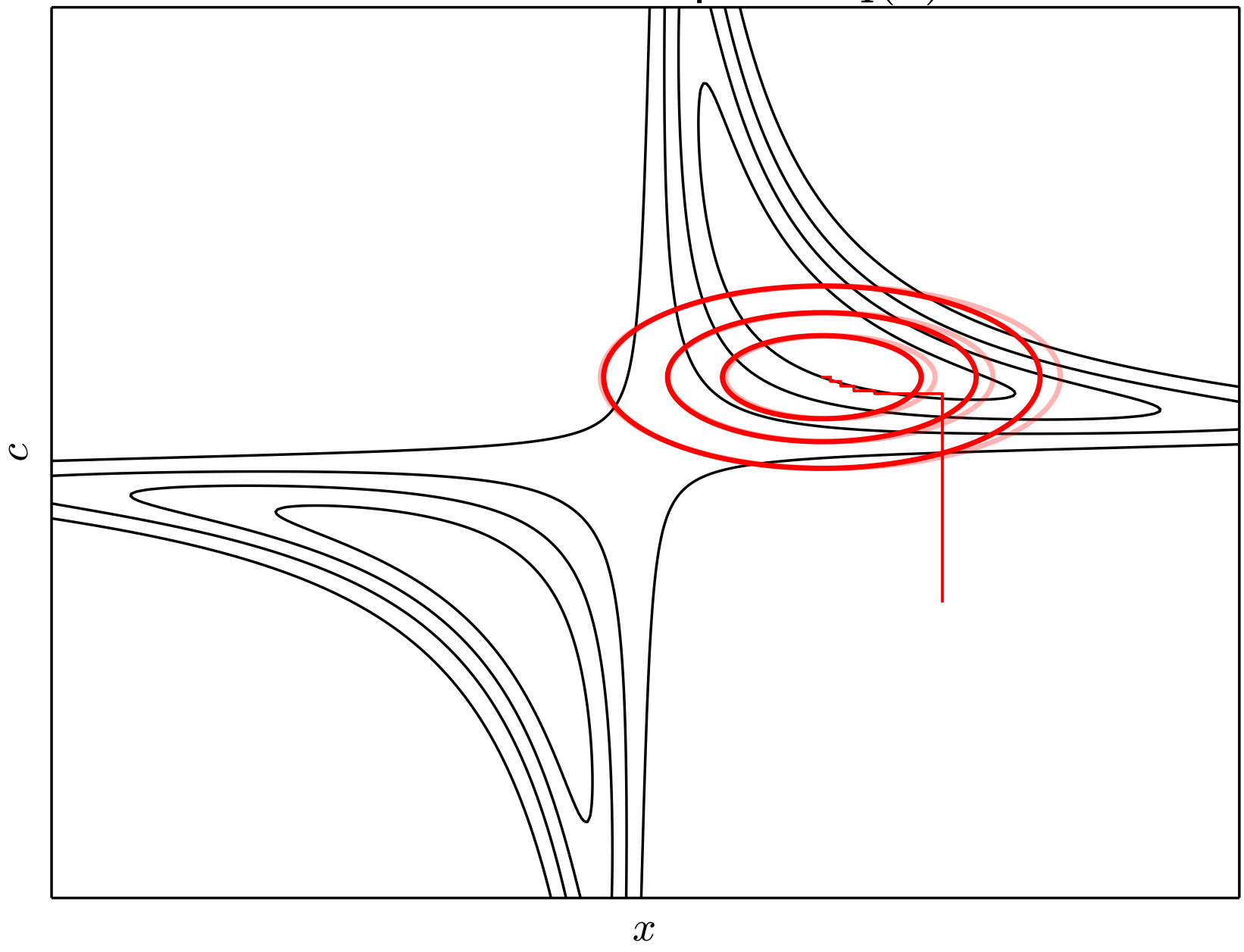
Iteration 4 - Update $q(x)$



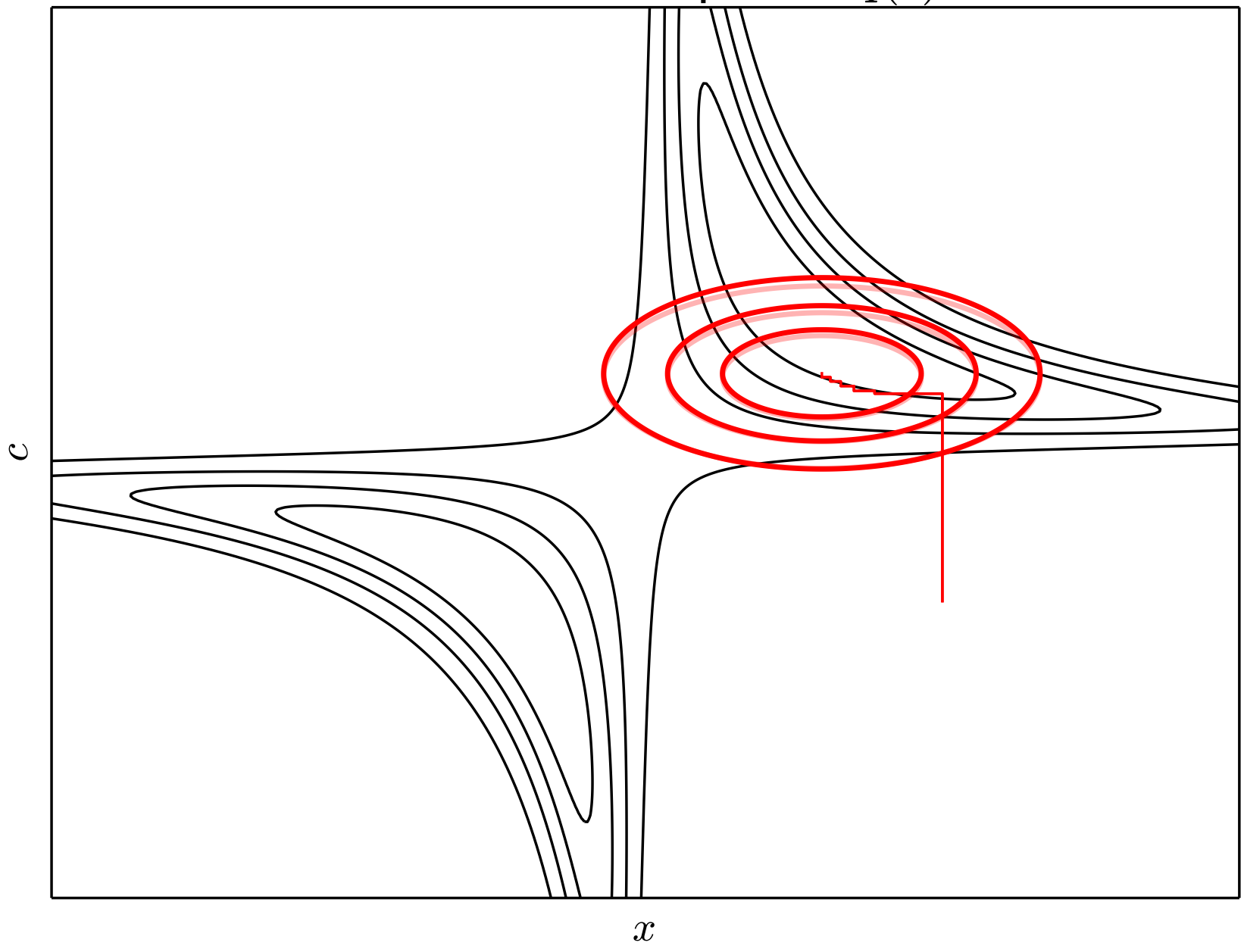
Iteration 5 - Update $q(c)$



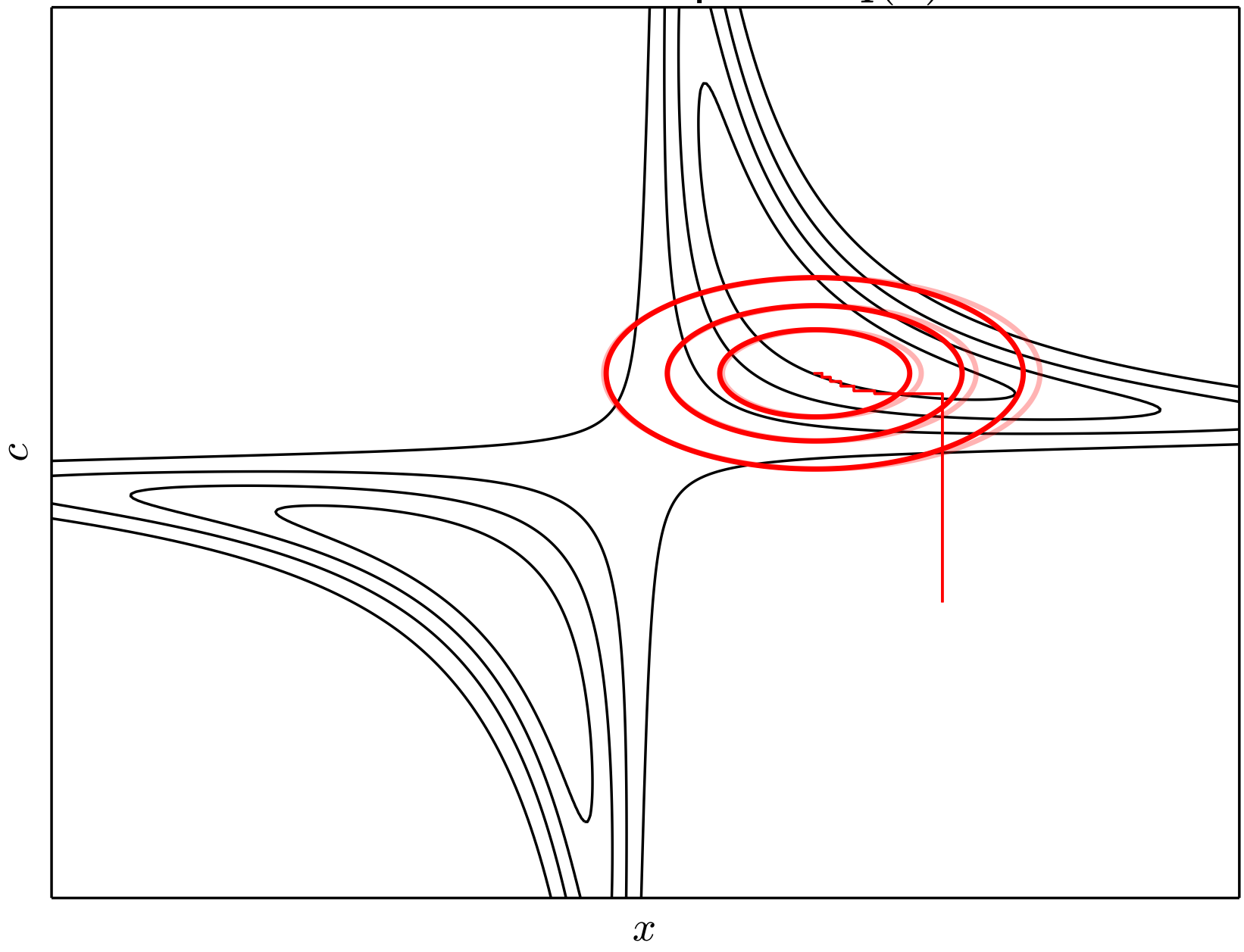
Iteration 5 - Update $q(x)$



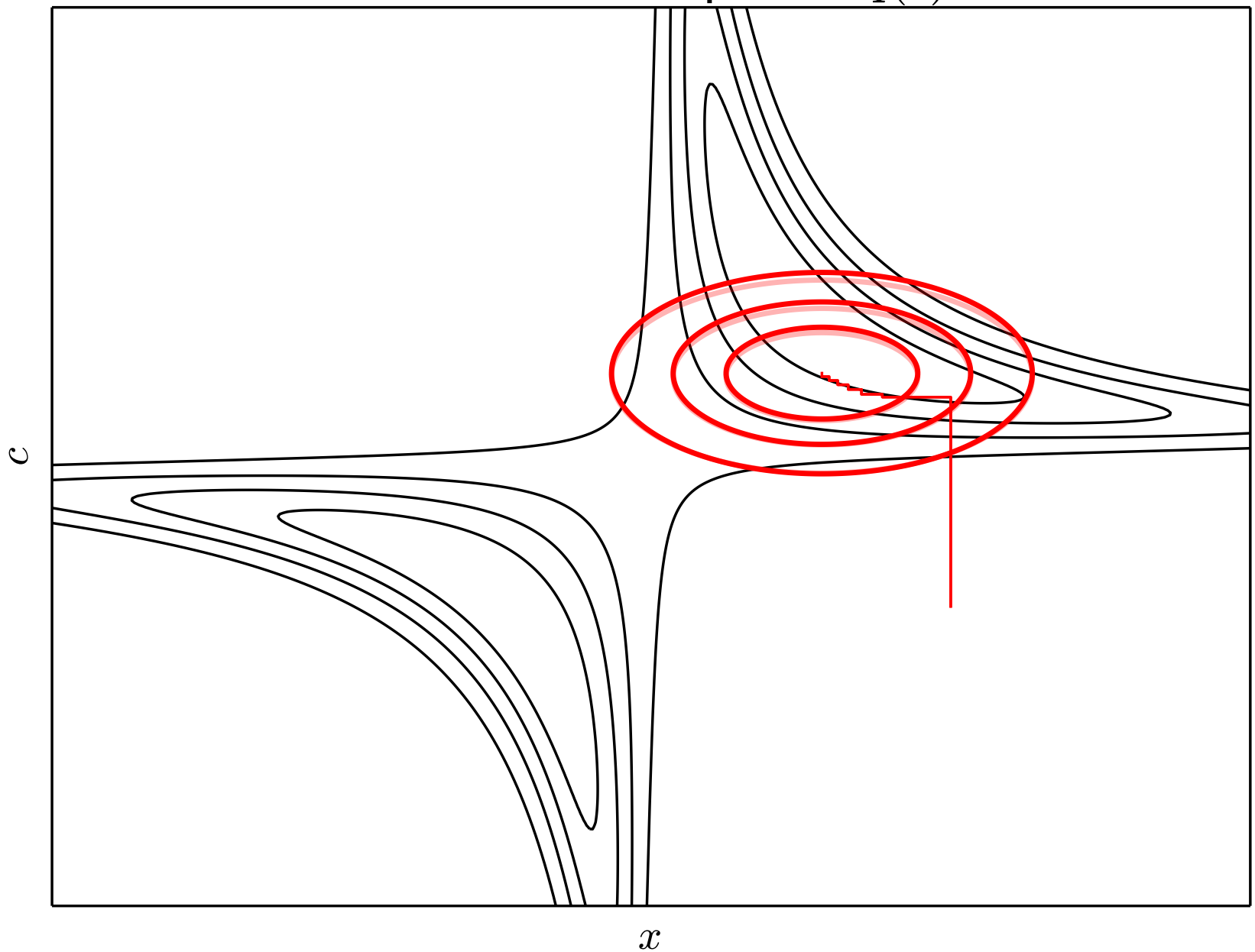
Iteration 6 - Update $q(c)$



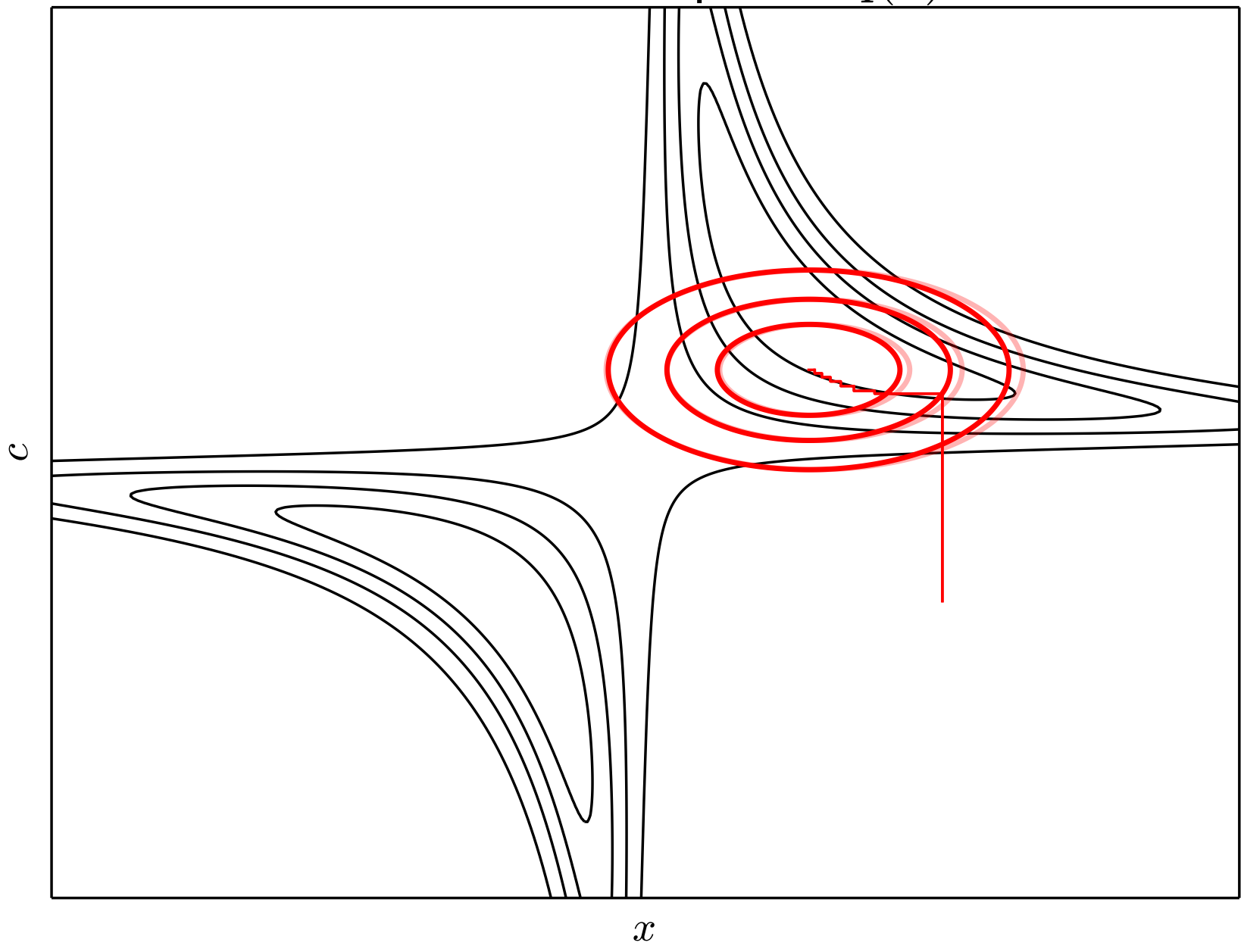
Iteration 6 - Update $q(x)$



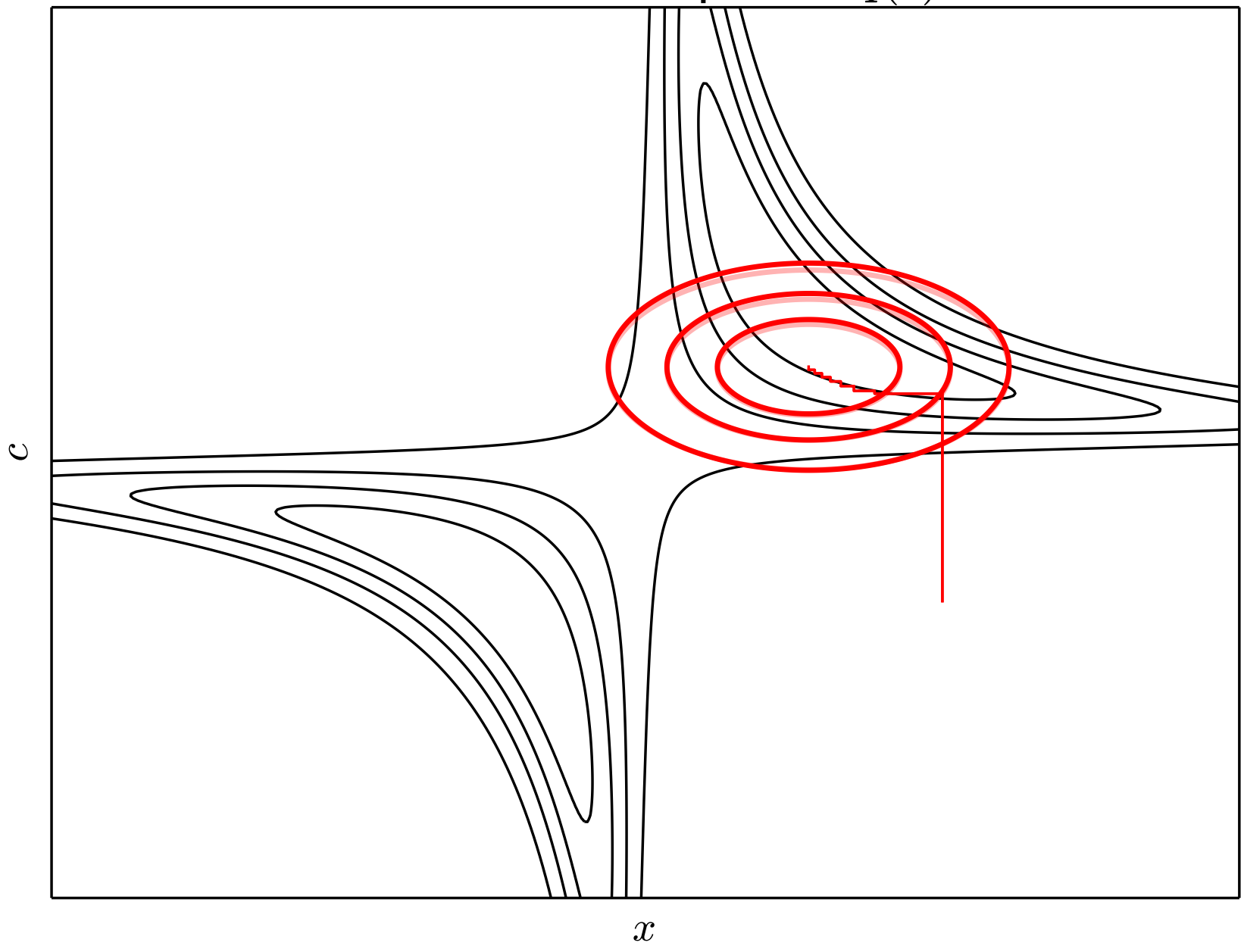
Iteration 7 - Update $q(c)$



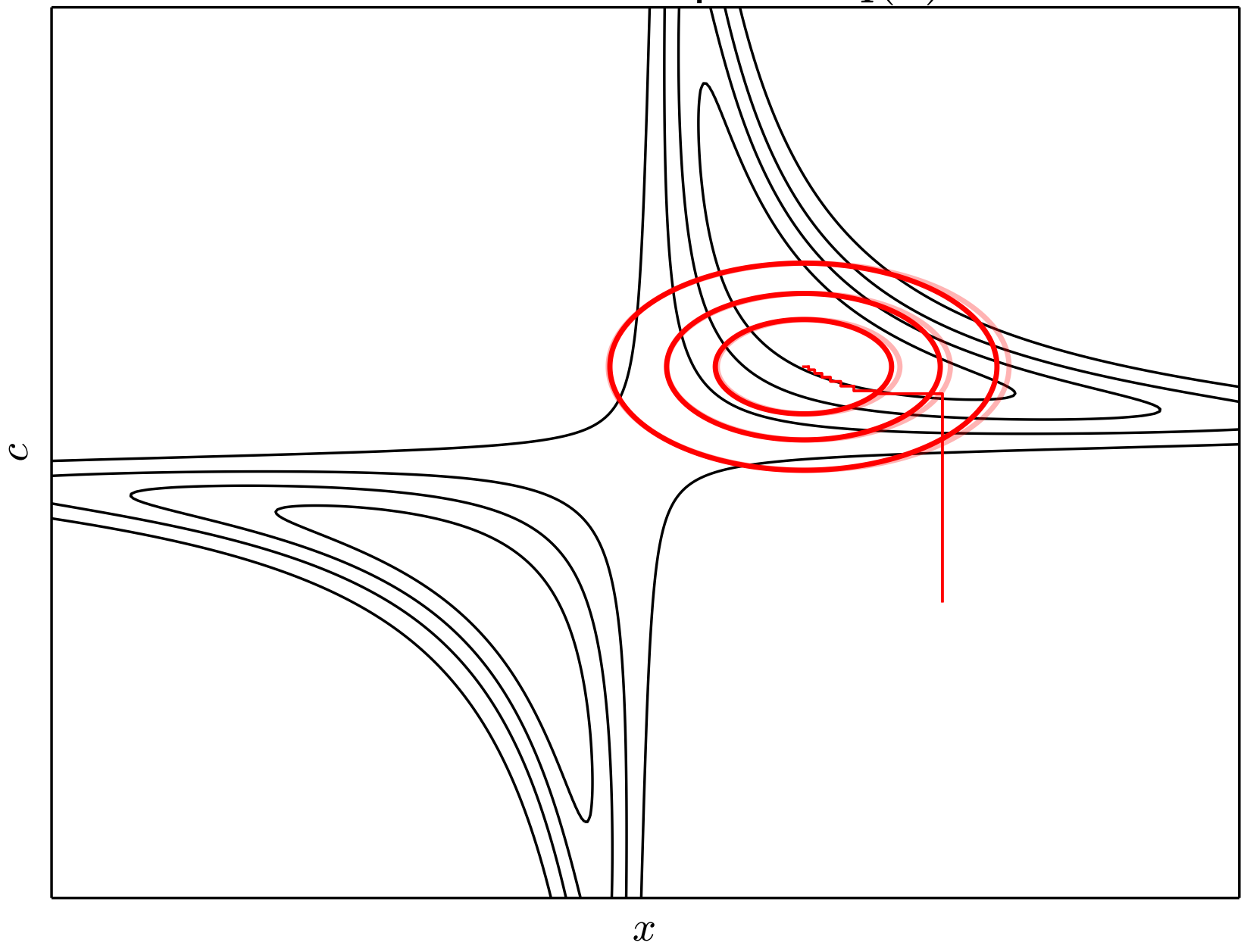
Iteration 7 - Update $q(x)$



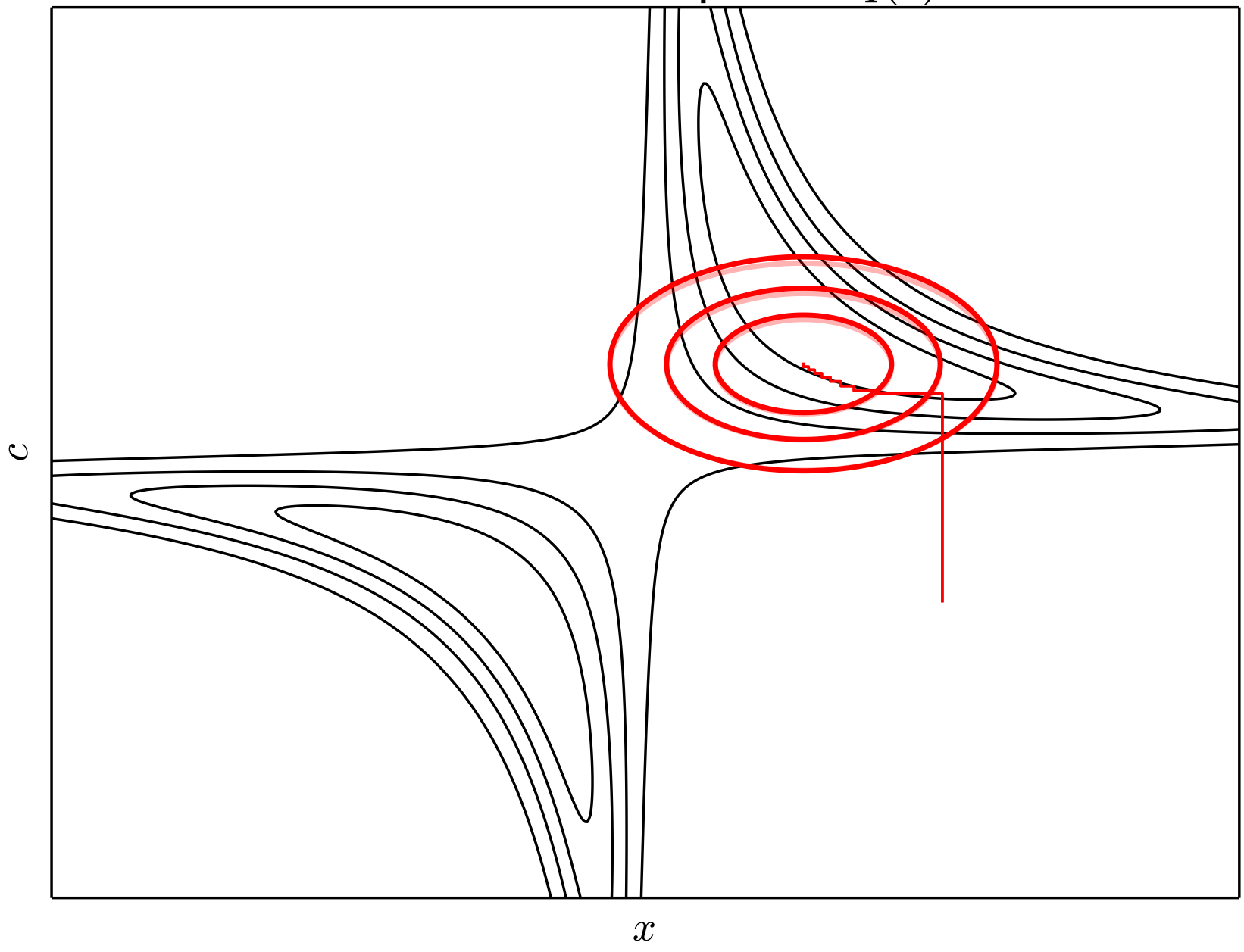
Iteration 8 - Update $q(c)$



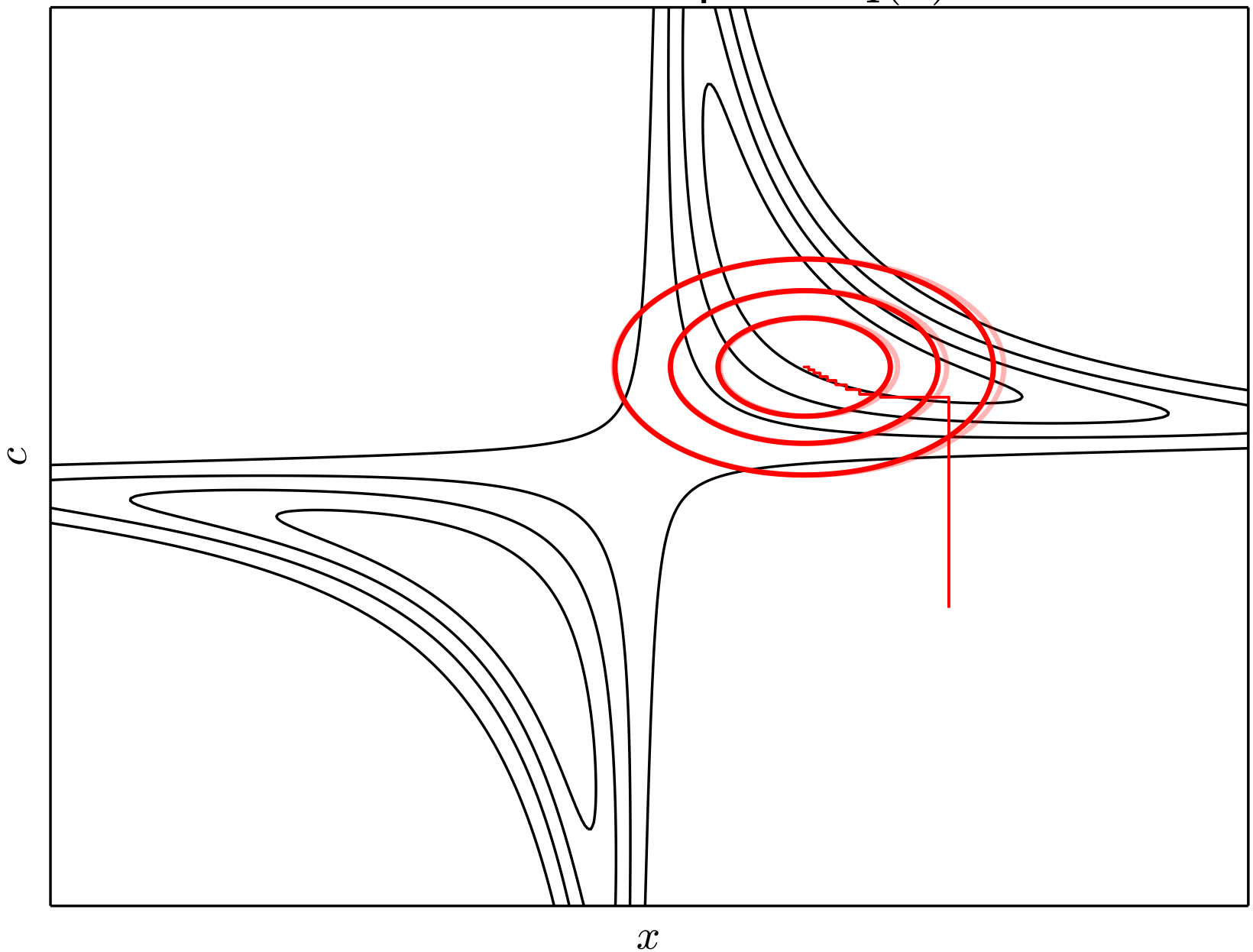
Iteration 8 - Update $q(x)$



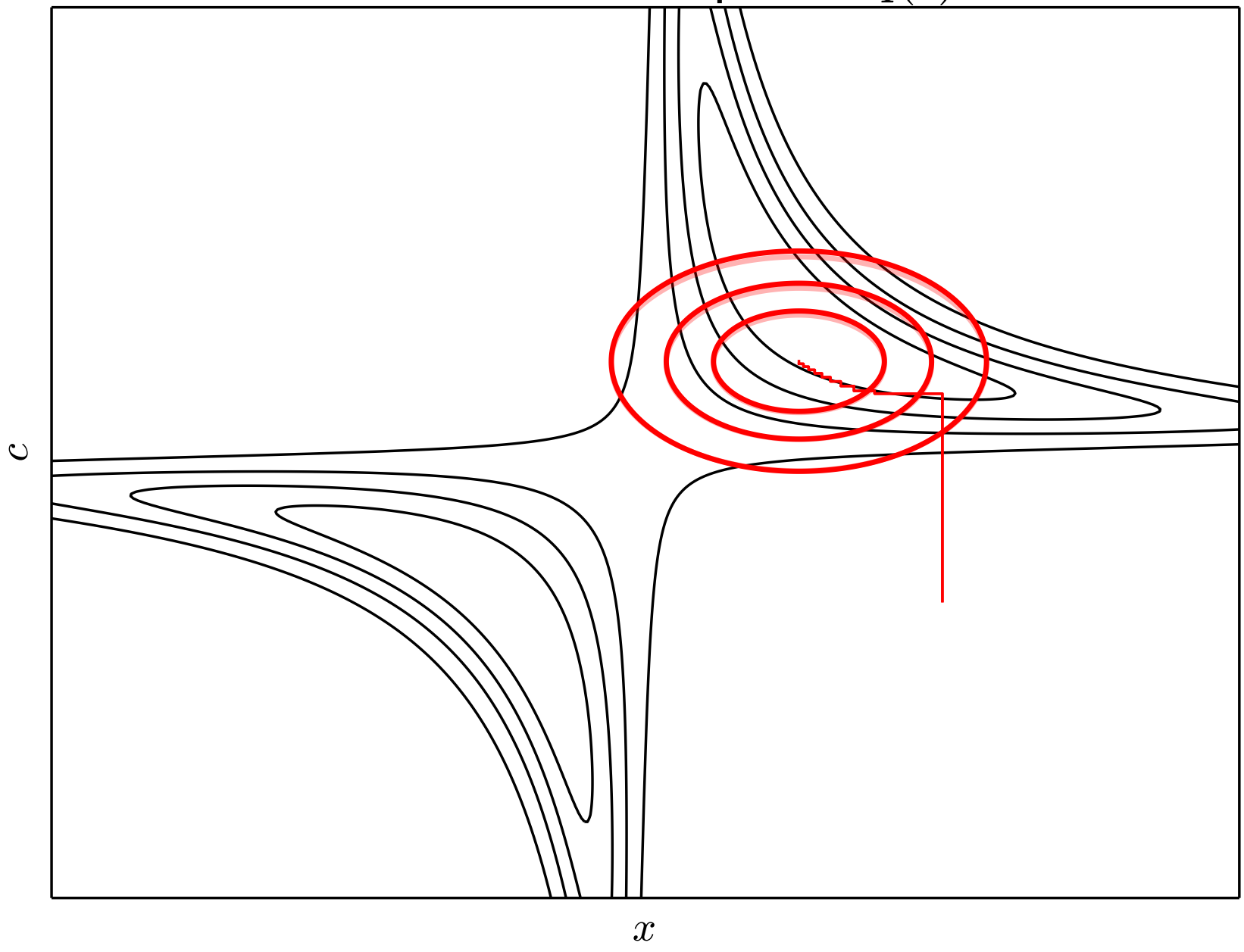
Iteration 9 - Update $q(c)$



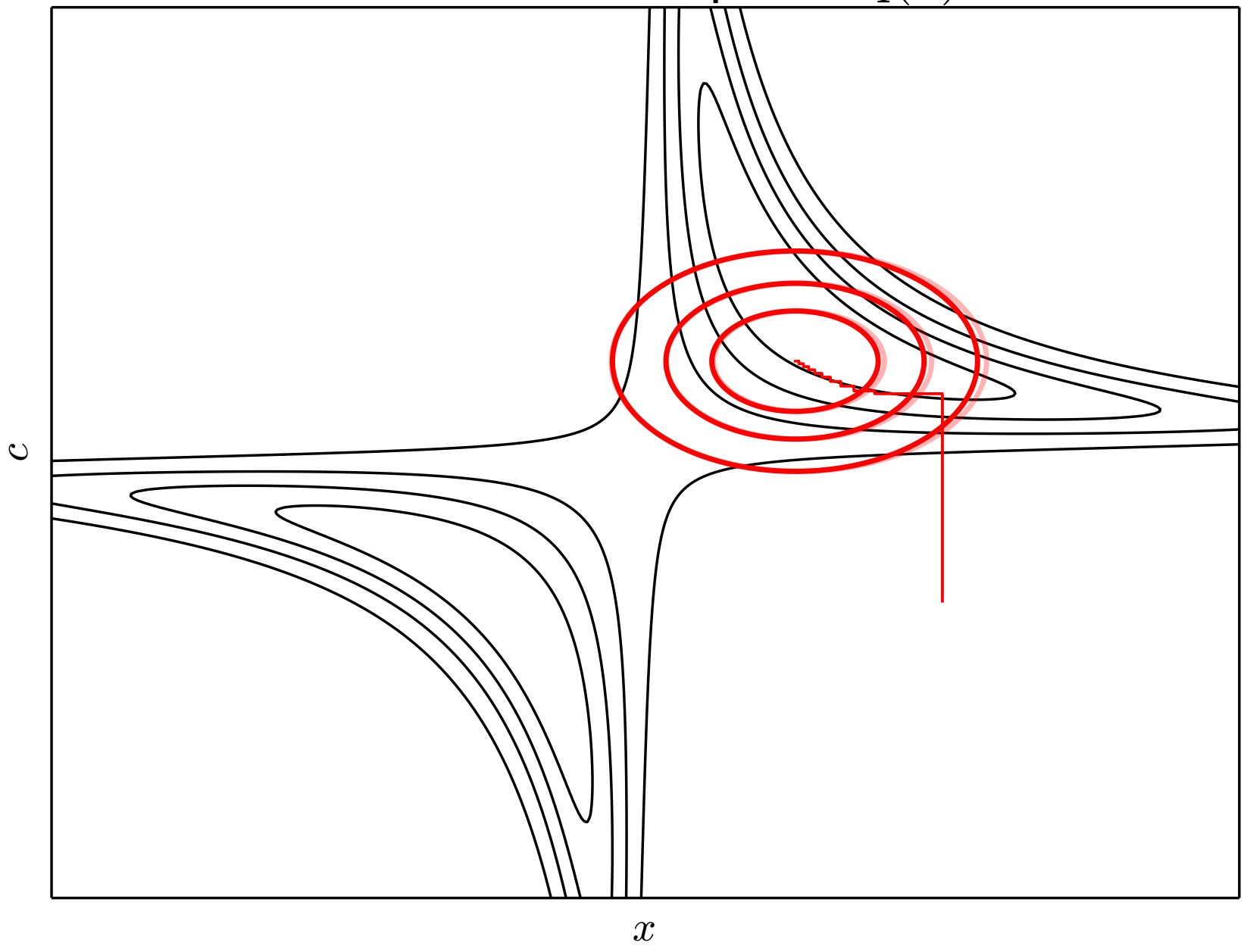
Iteration 9 - Update $q(x)$



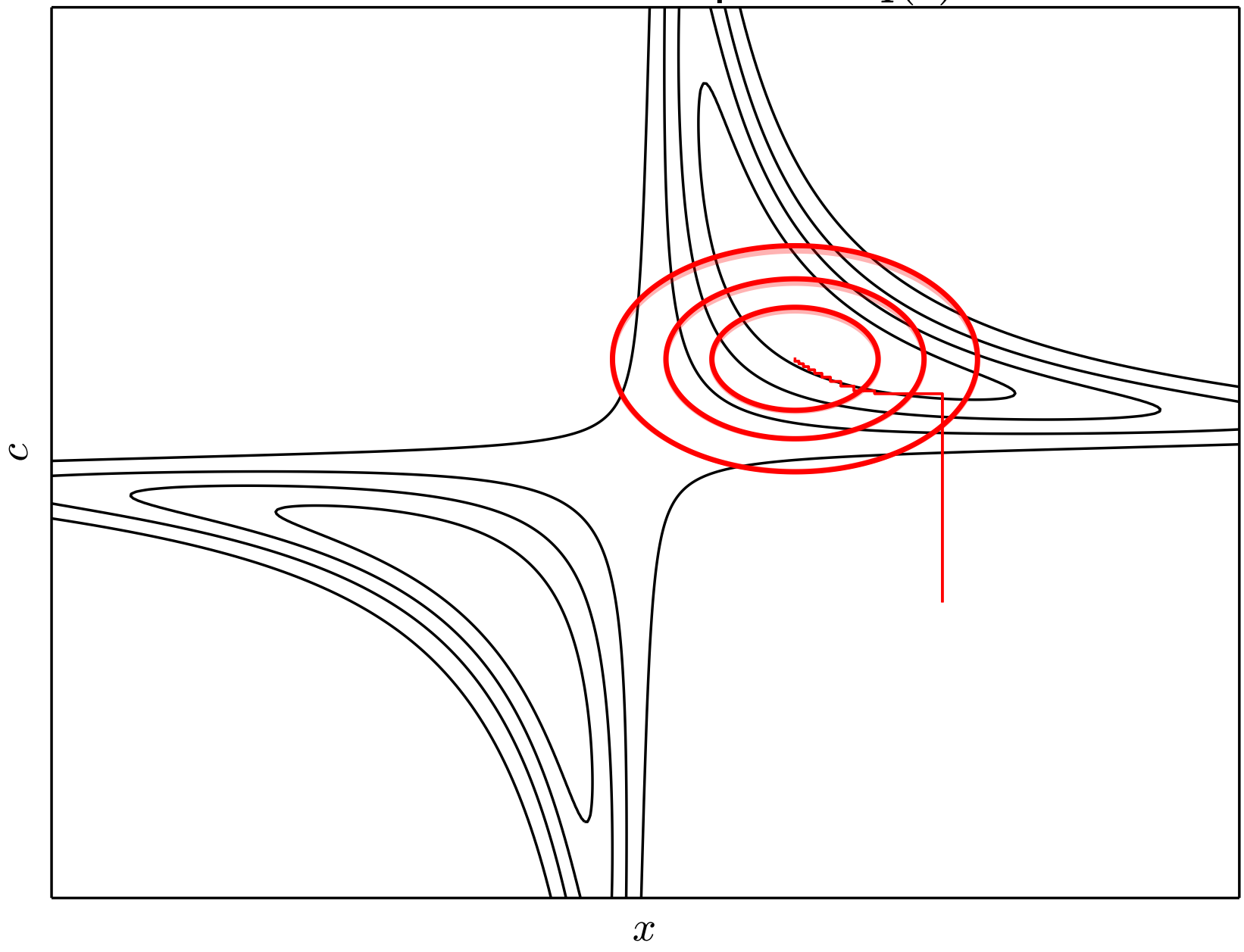
Iteration 10 - Update $q(c)$



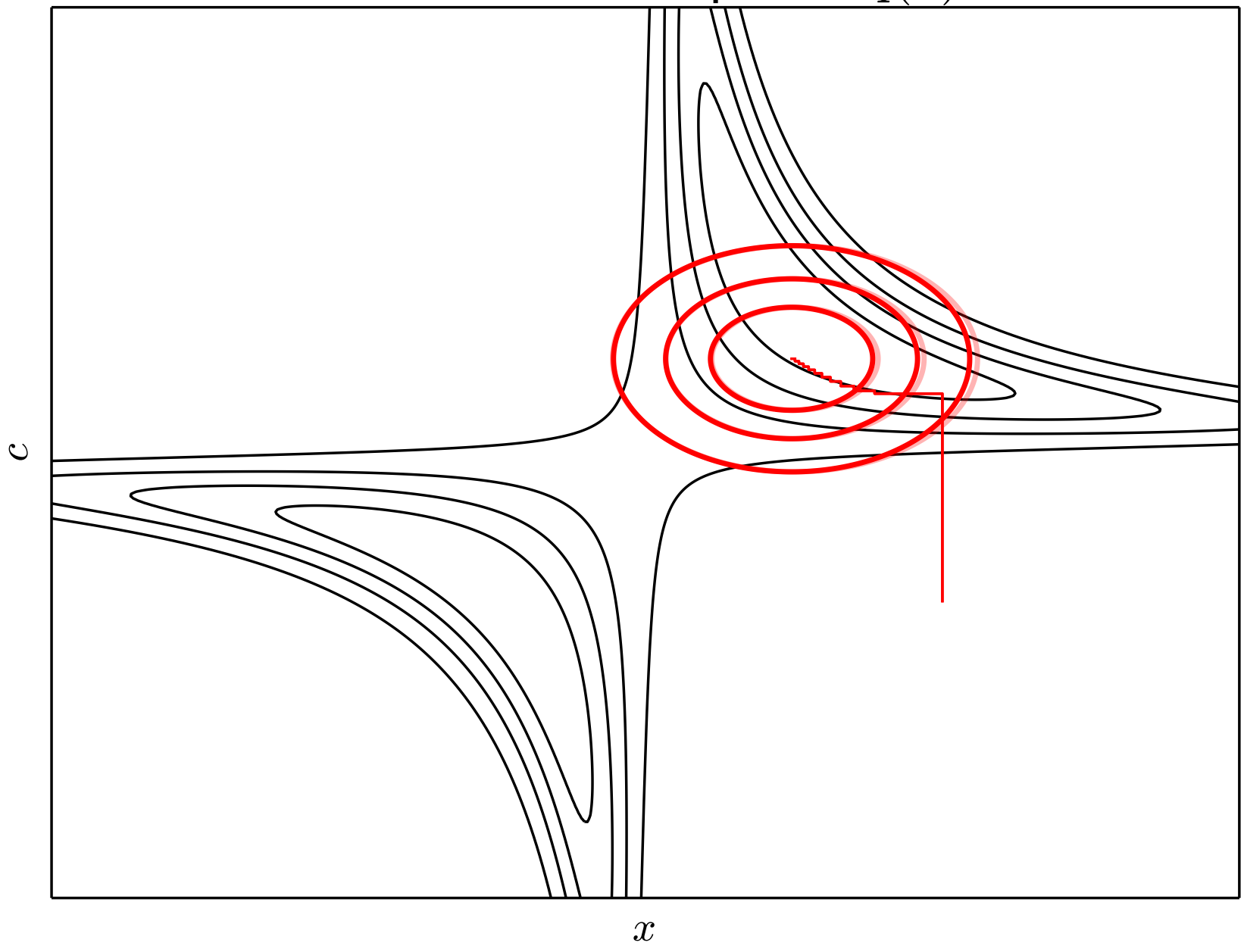
Iteration 10 - Update $q(x)$



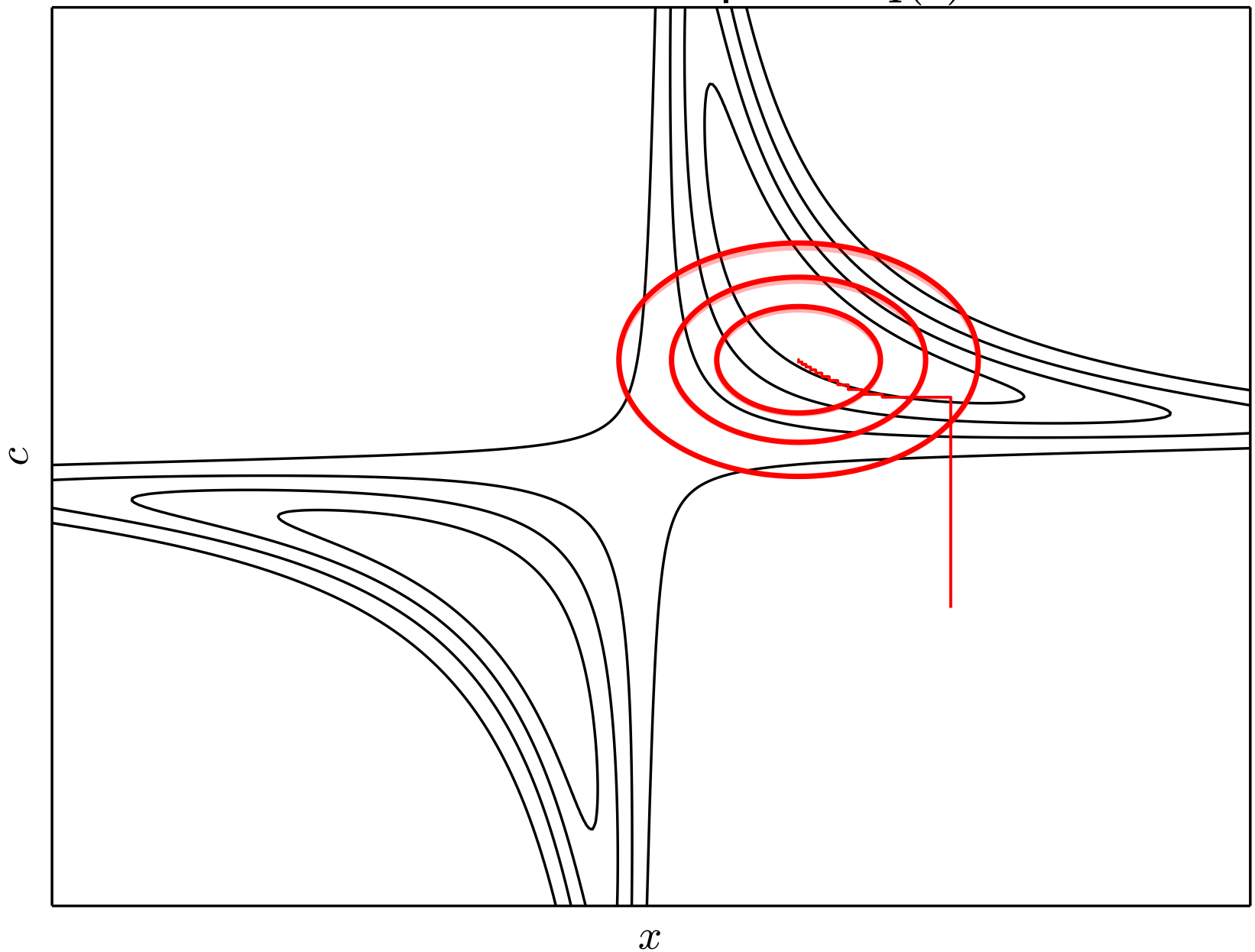
Iteration 11 - Update $q(c)$



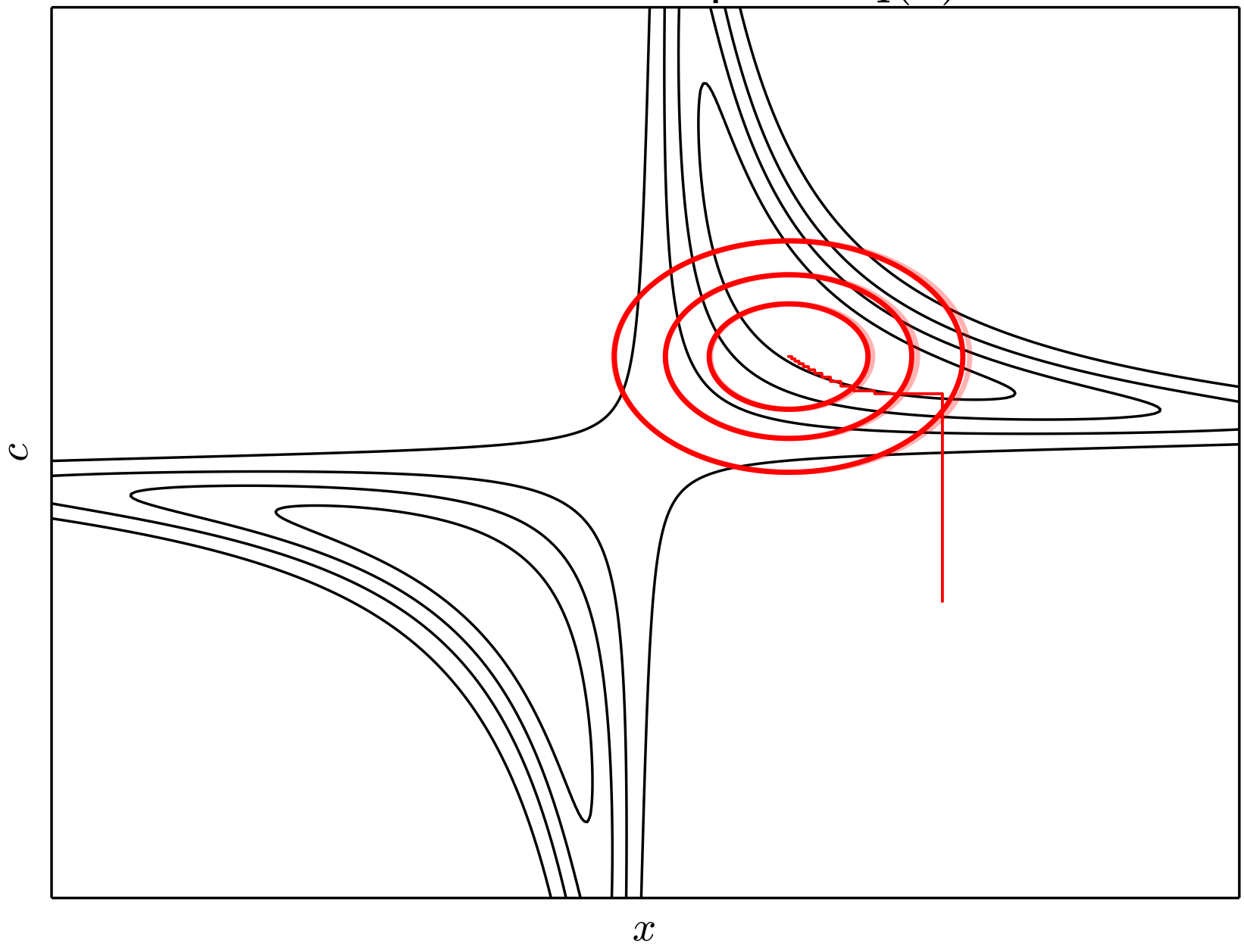
Iteration 11 - Update $q(x)$



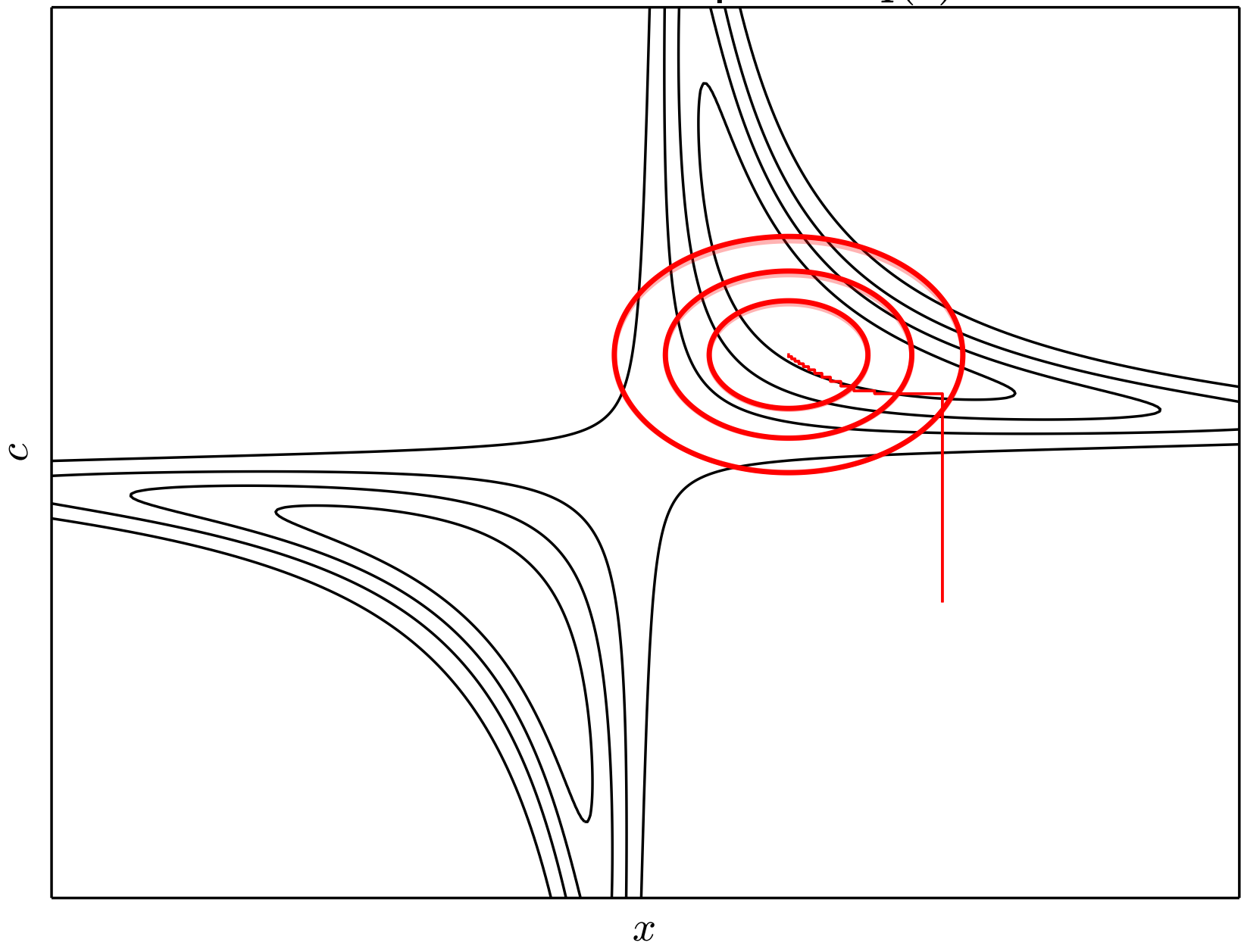
Iteration 12 - Update $q(c)$



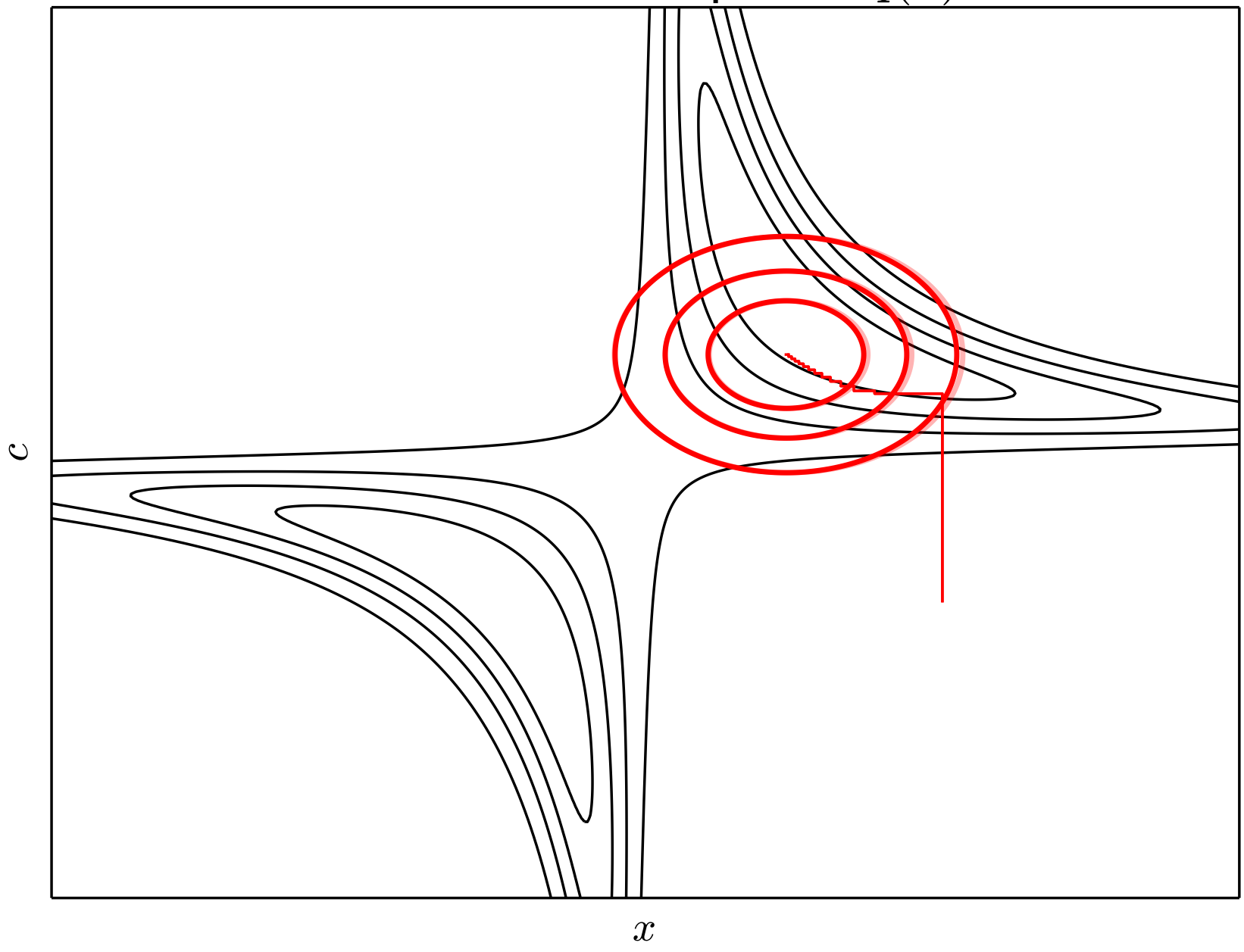
Iteration 12 - Update $q(x)$



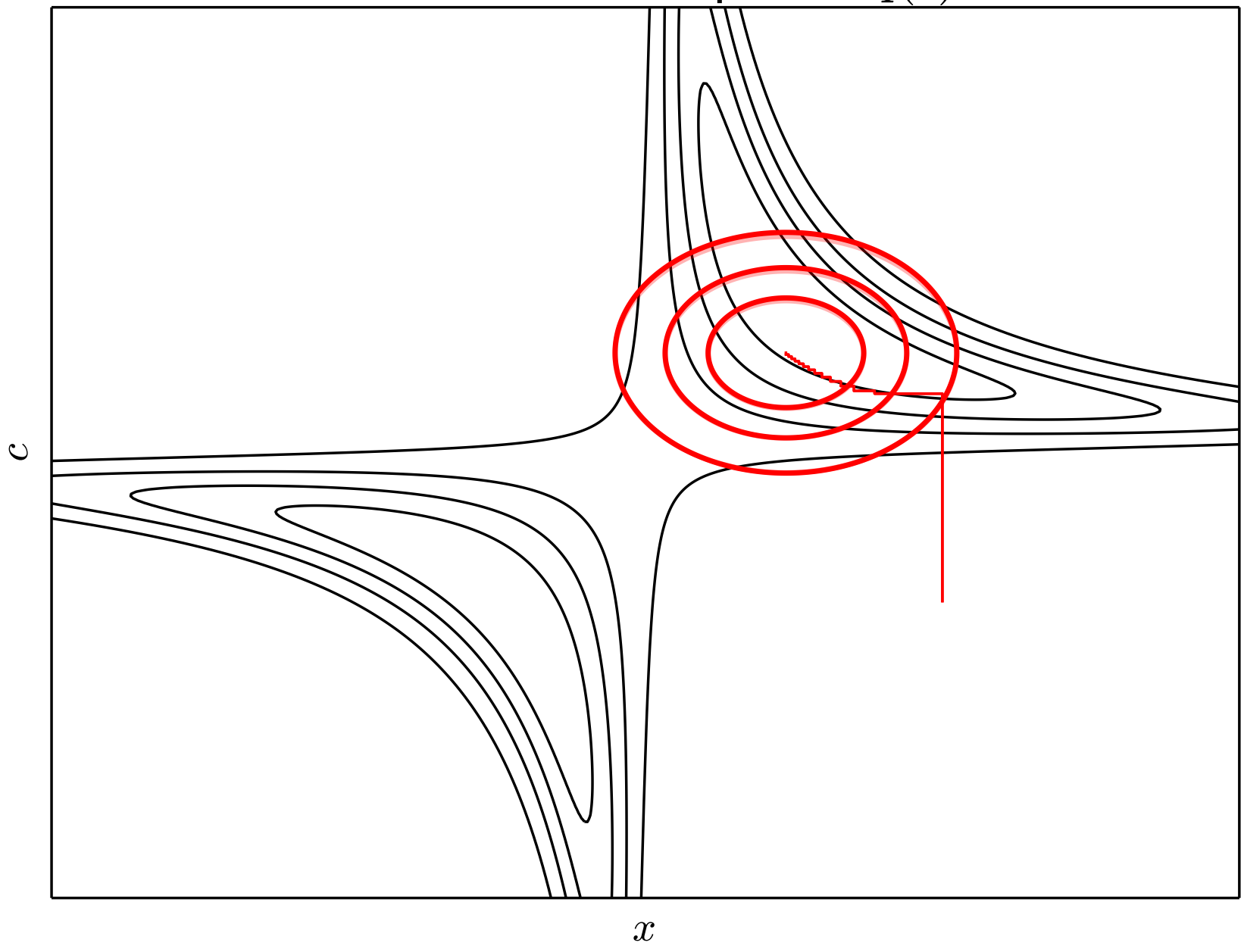
Iteration 13 - Update $q(c)$



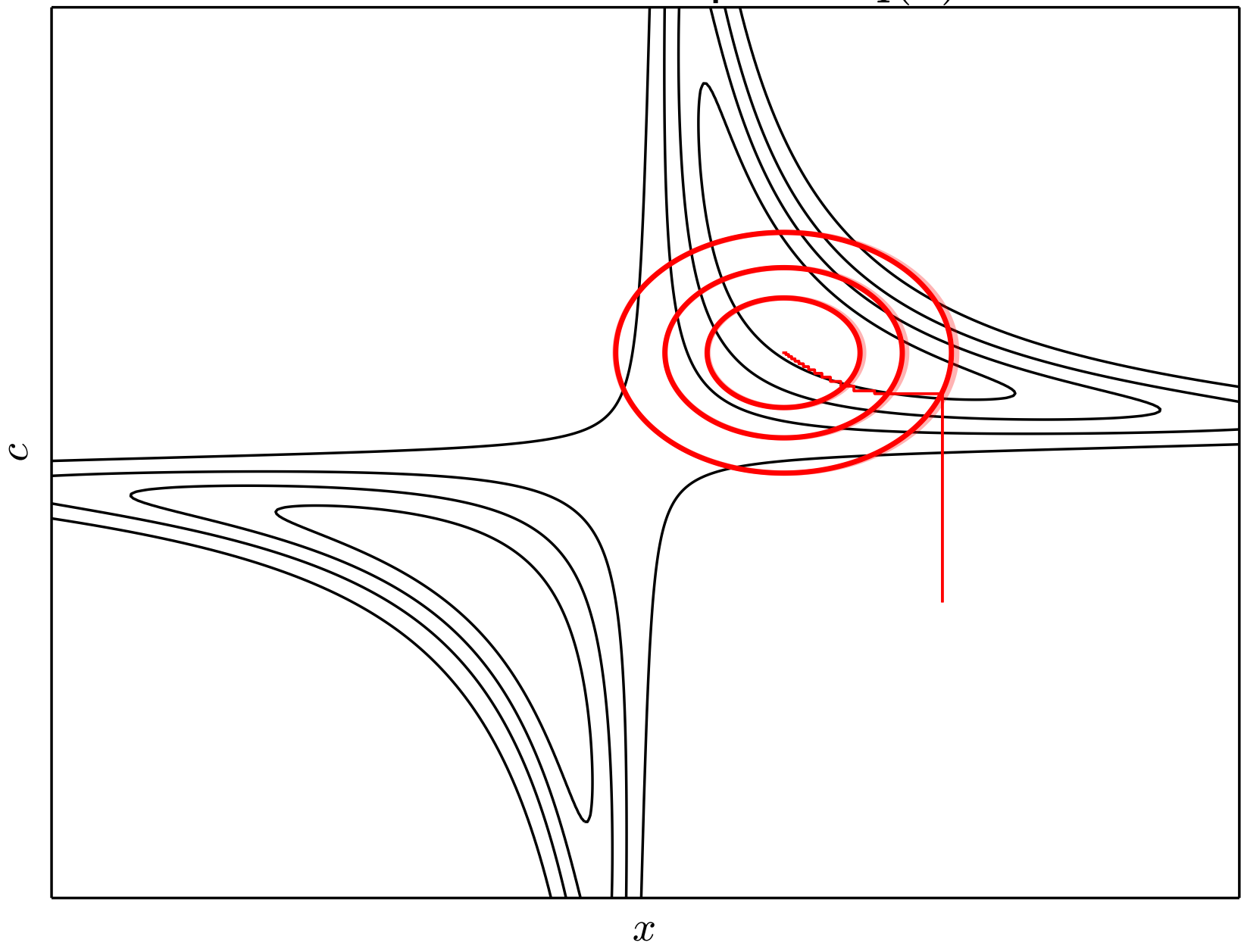
Iteration 13 - Update $q(x)$



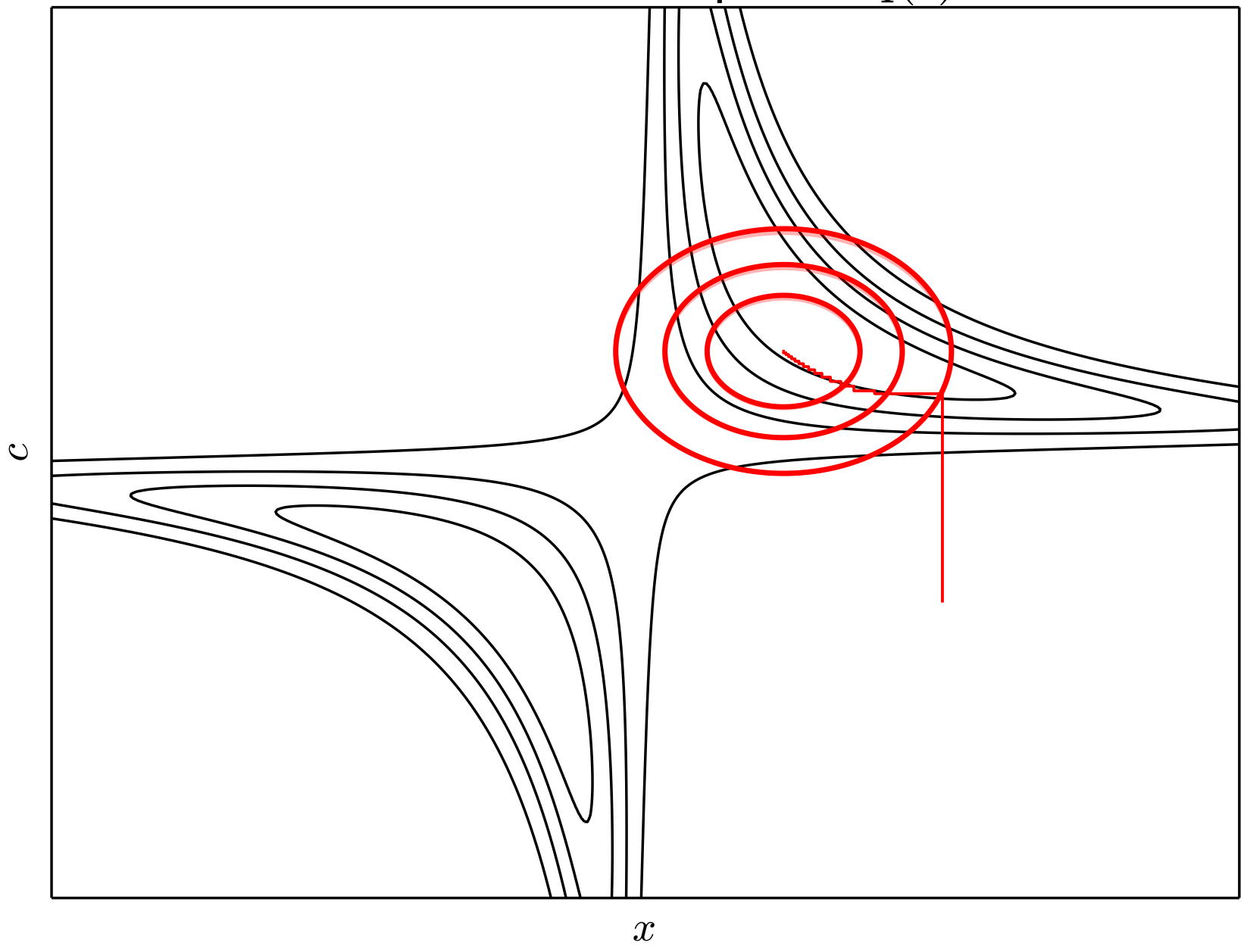
Iteration 14 - Update $q(c)$



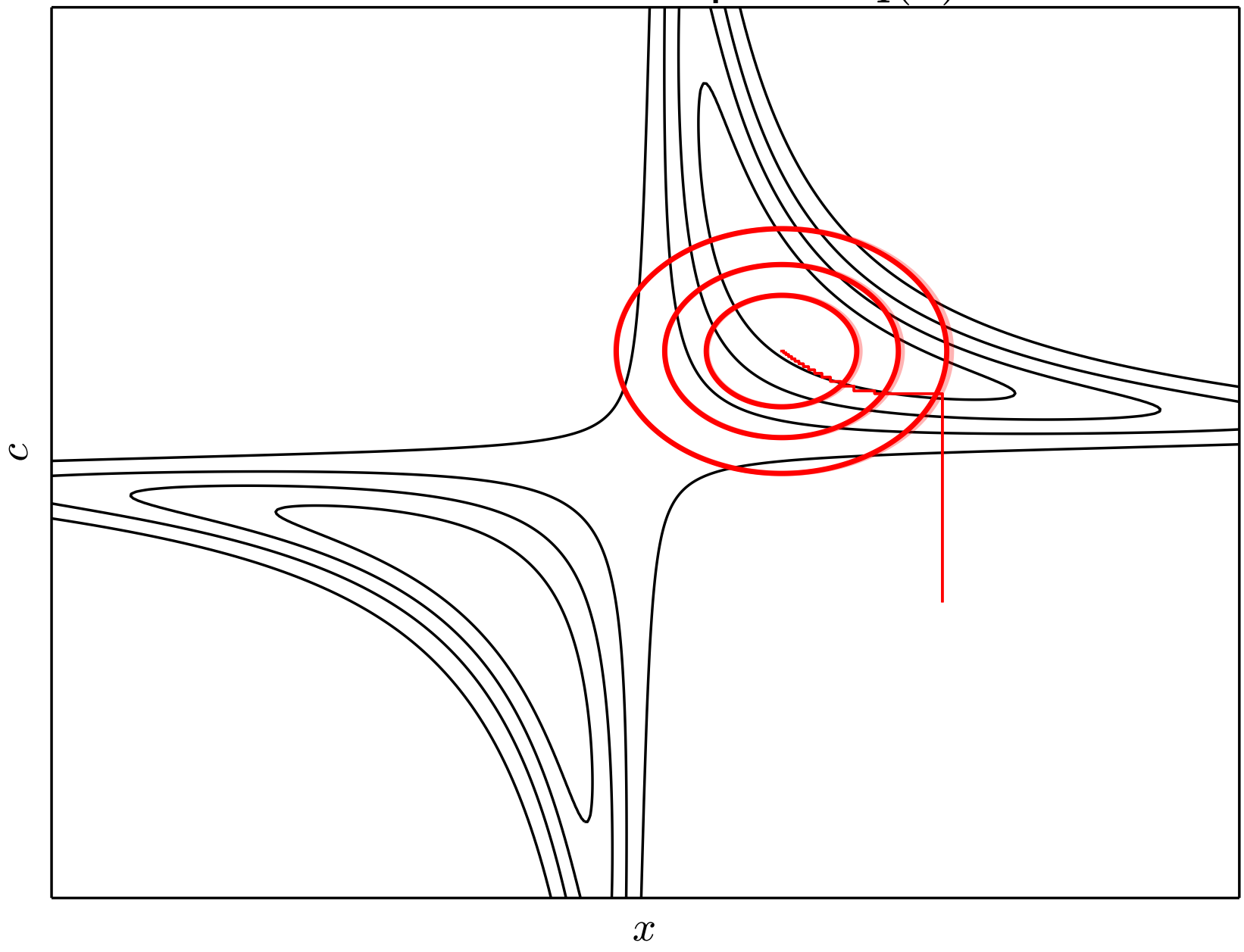
Iteration 14 - Update $q(x)$



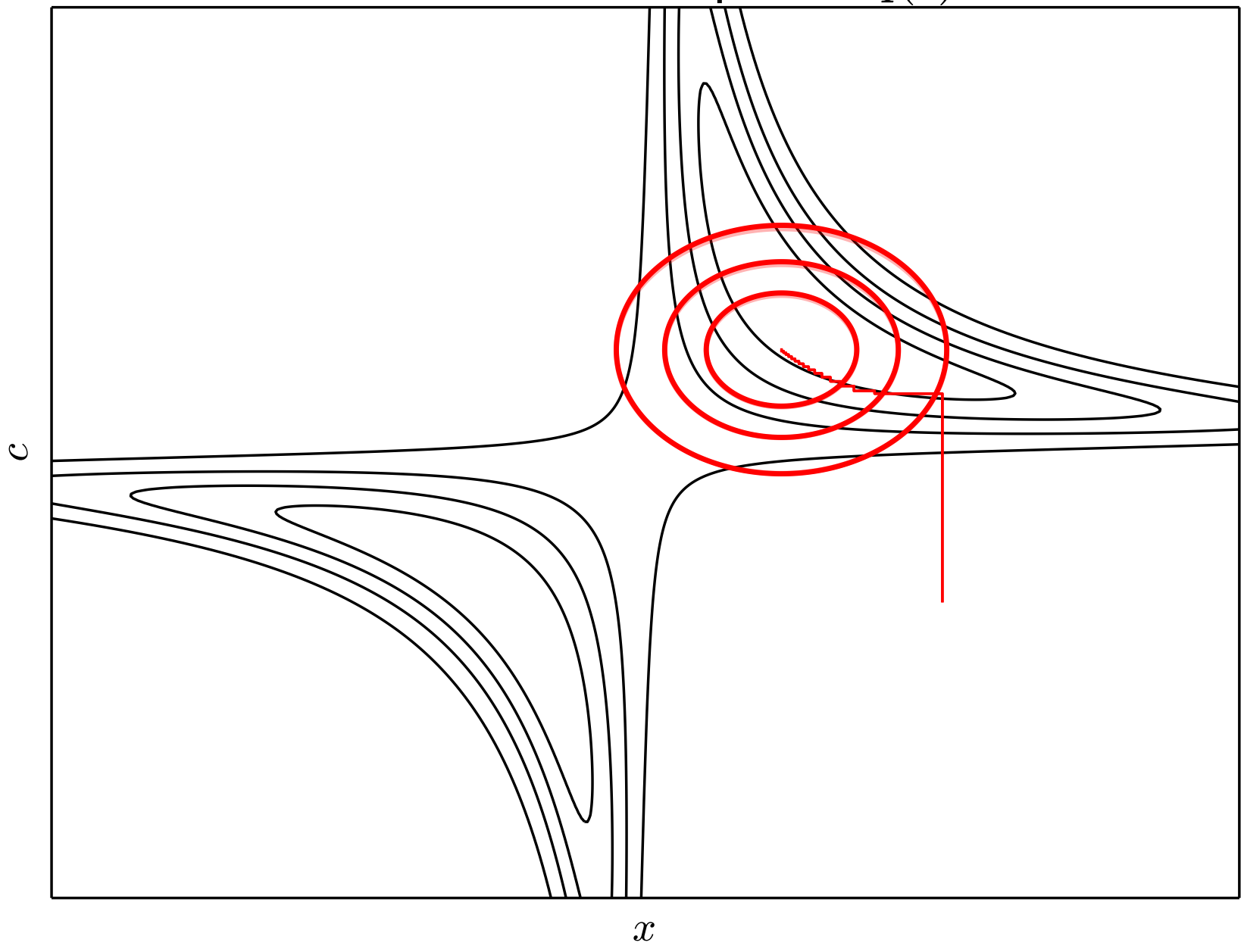
Iteration 15 - Update $q(c)$



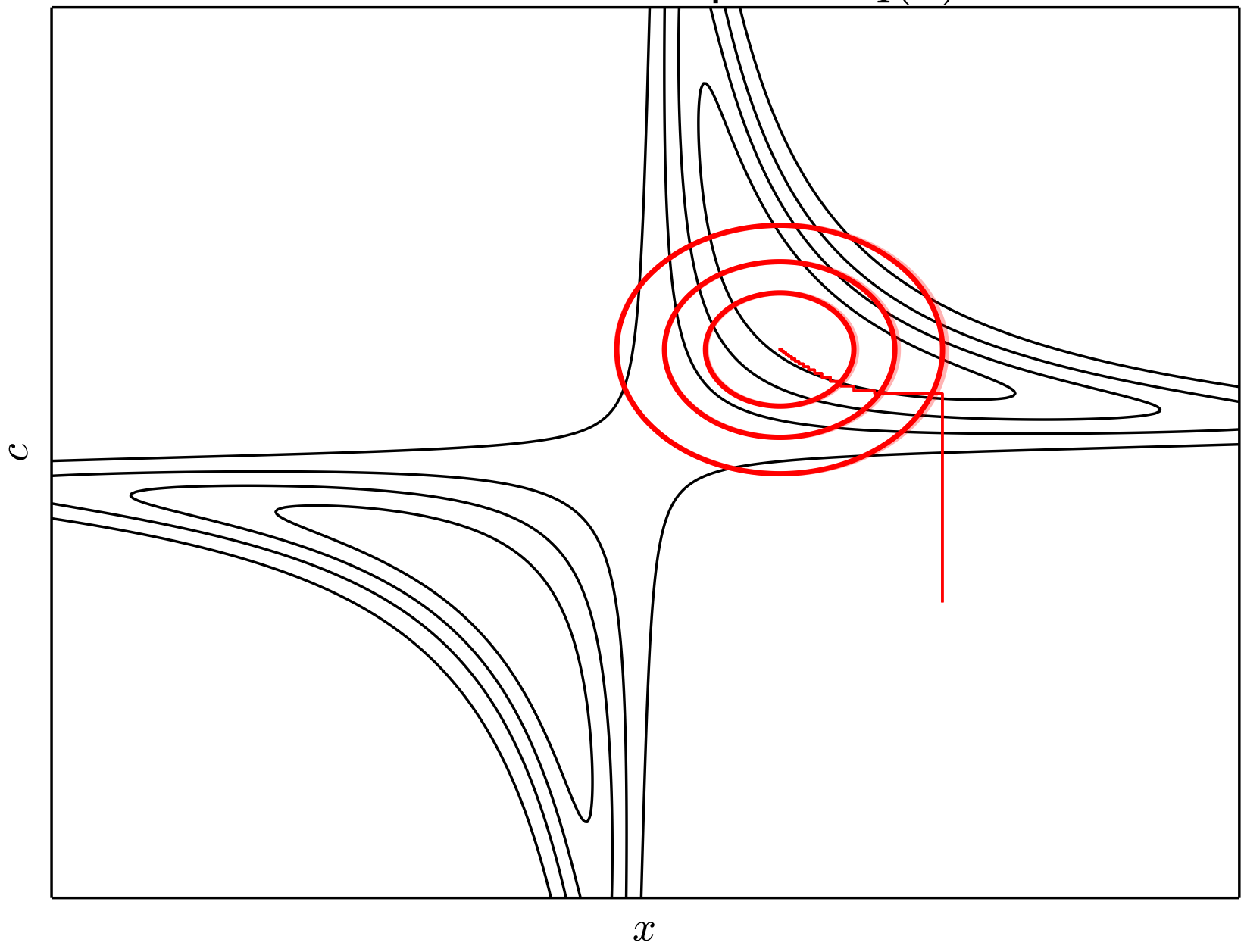
Iteration 15 - Update $q(x)$



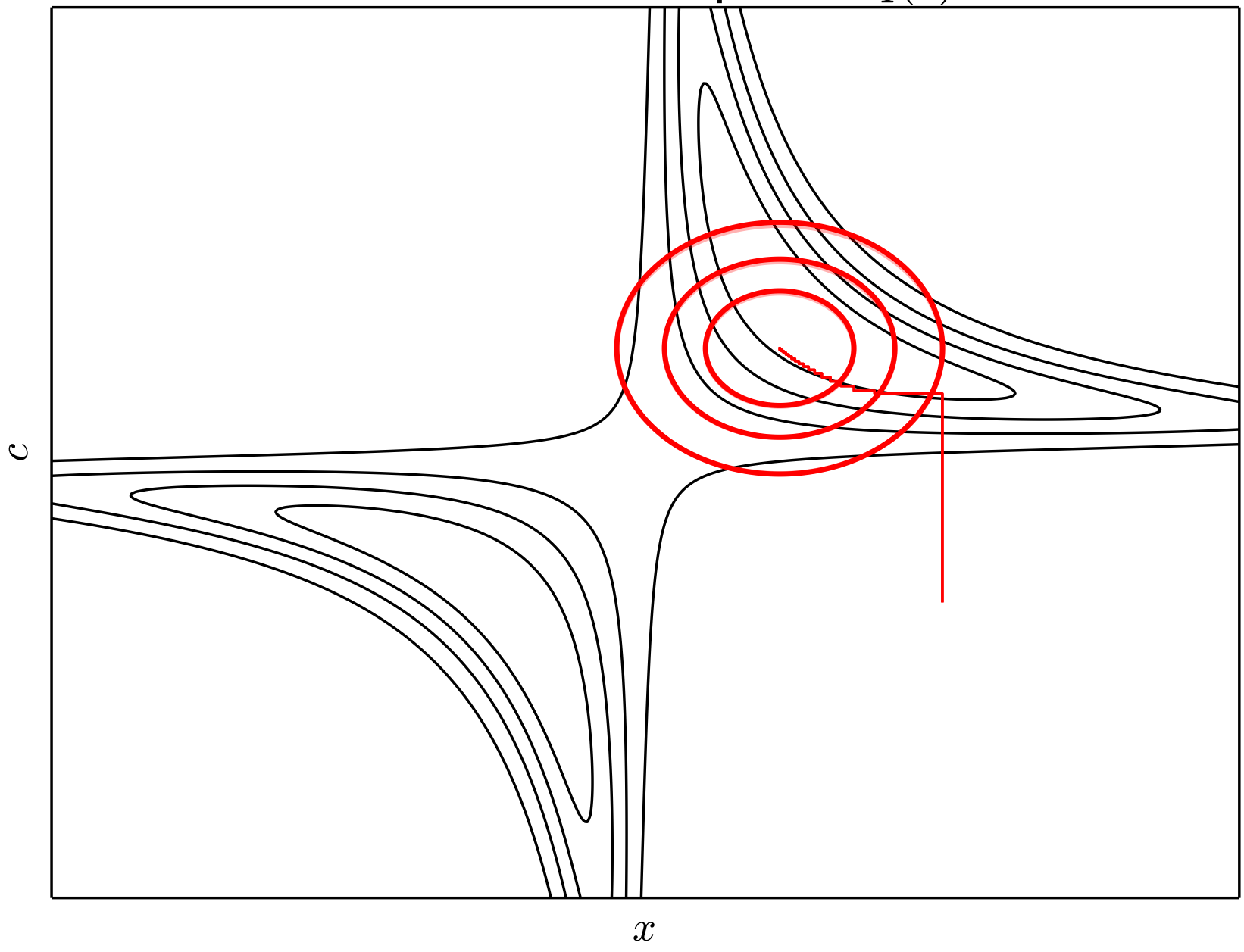
Iteration 16 - Update $q(c)$



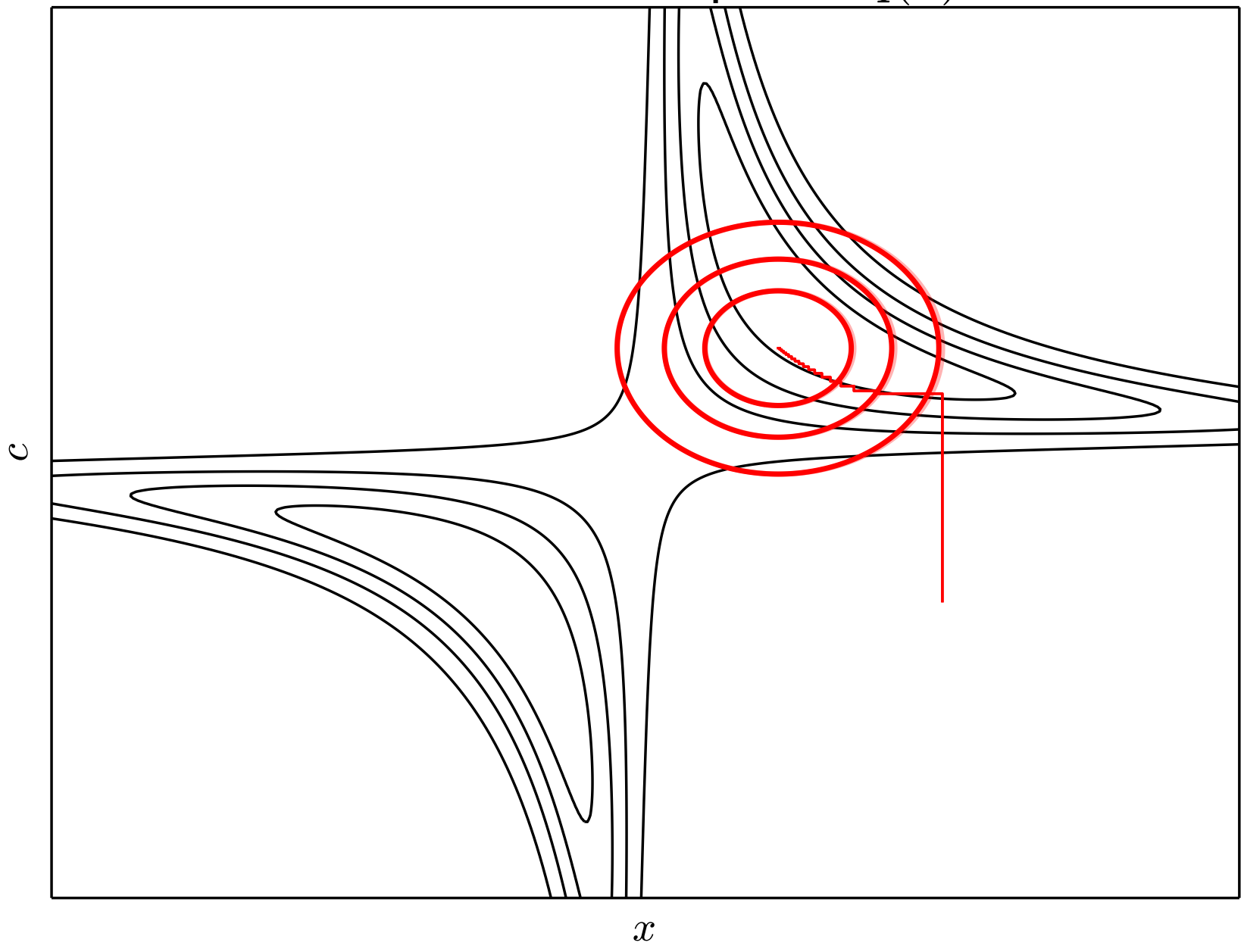
Iteration 16 - Update $q(x)$



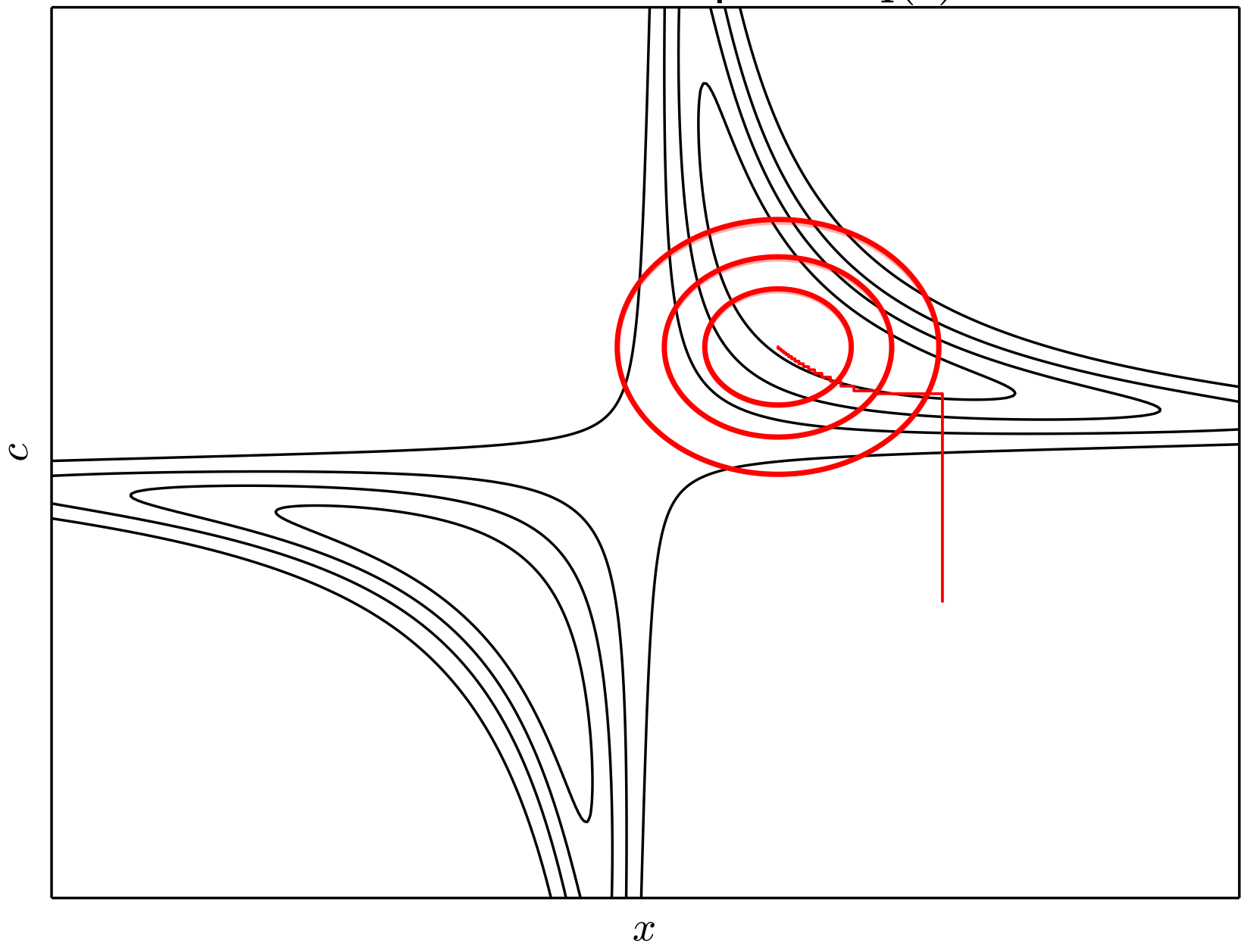
Iteration 17 - Update $q(c)$



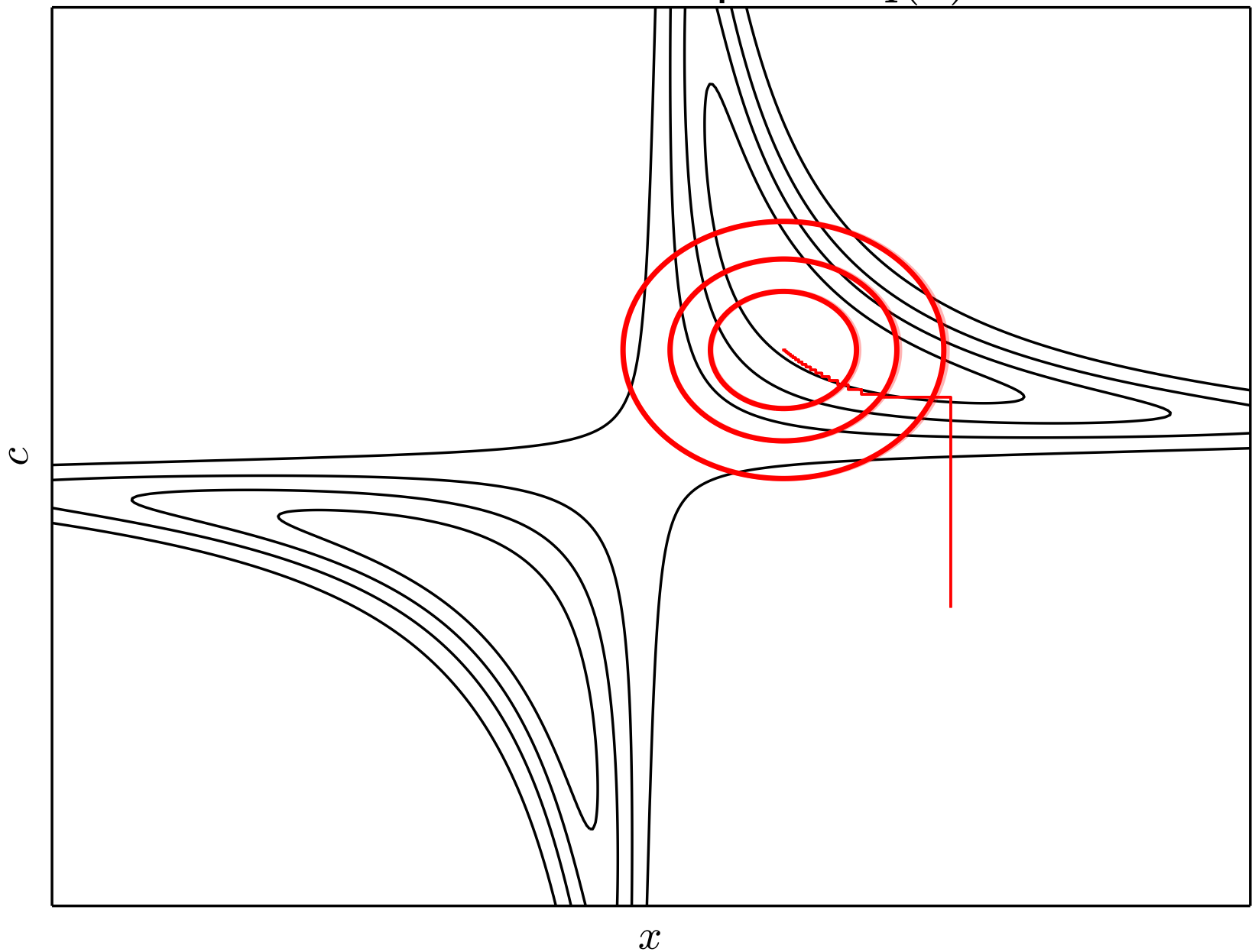
Iteration 17 - Update $q(x)$



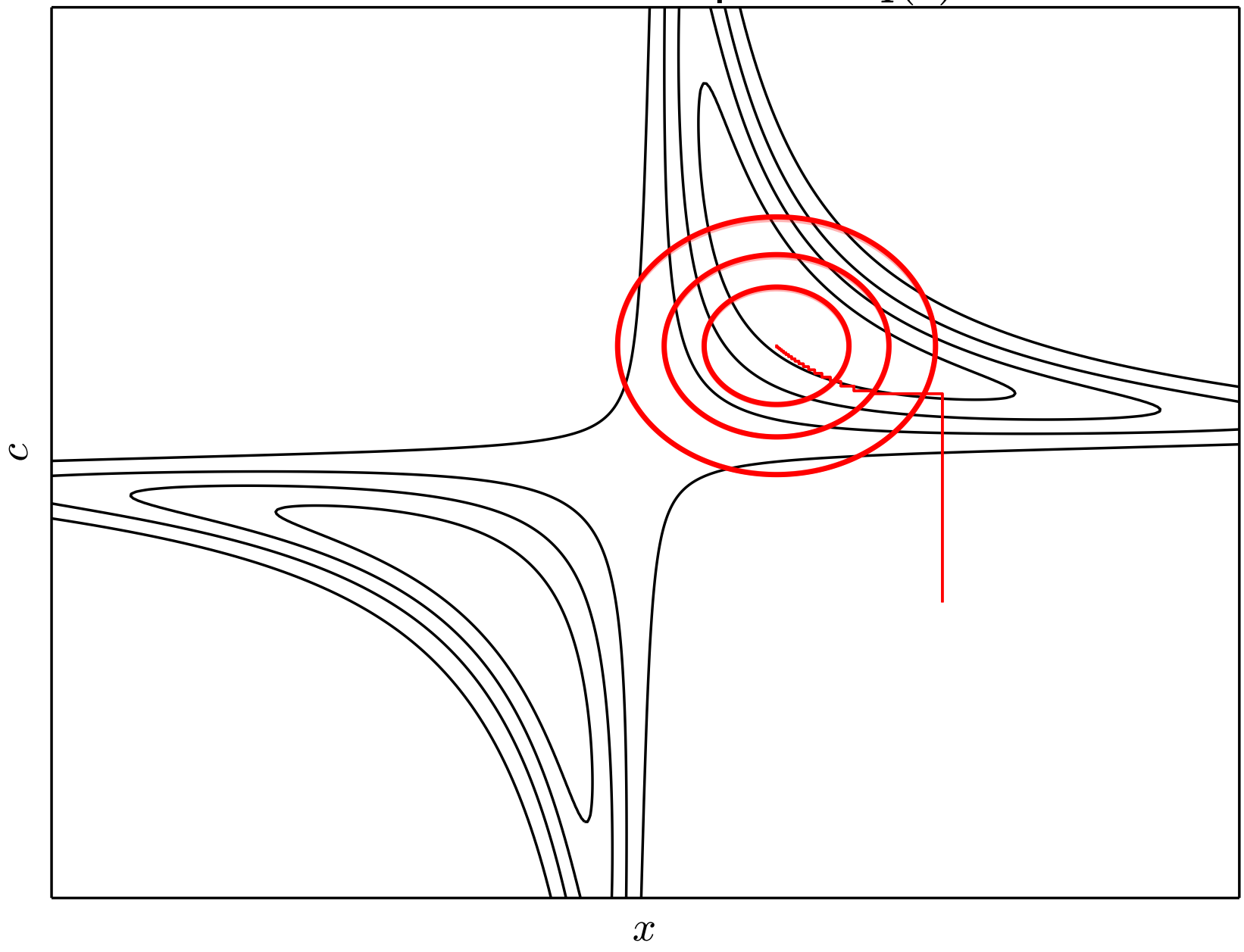
Iteration 18 - Update $q(c)$



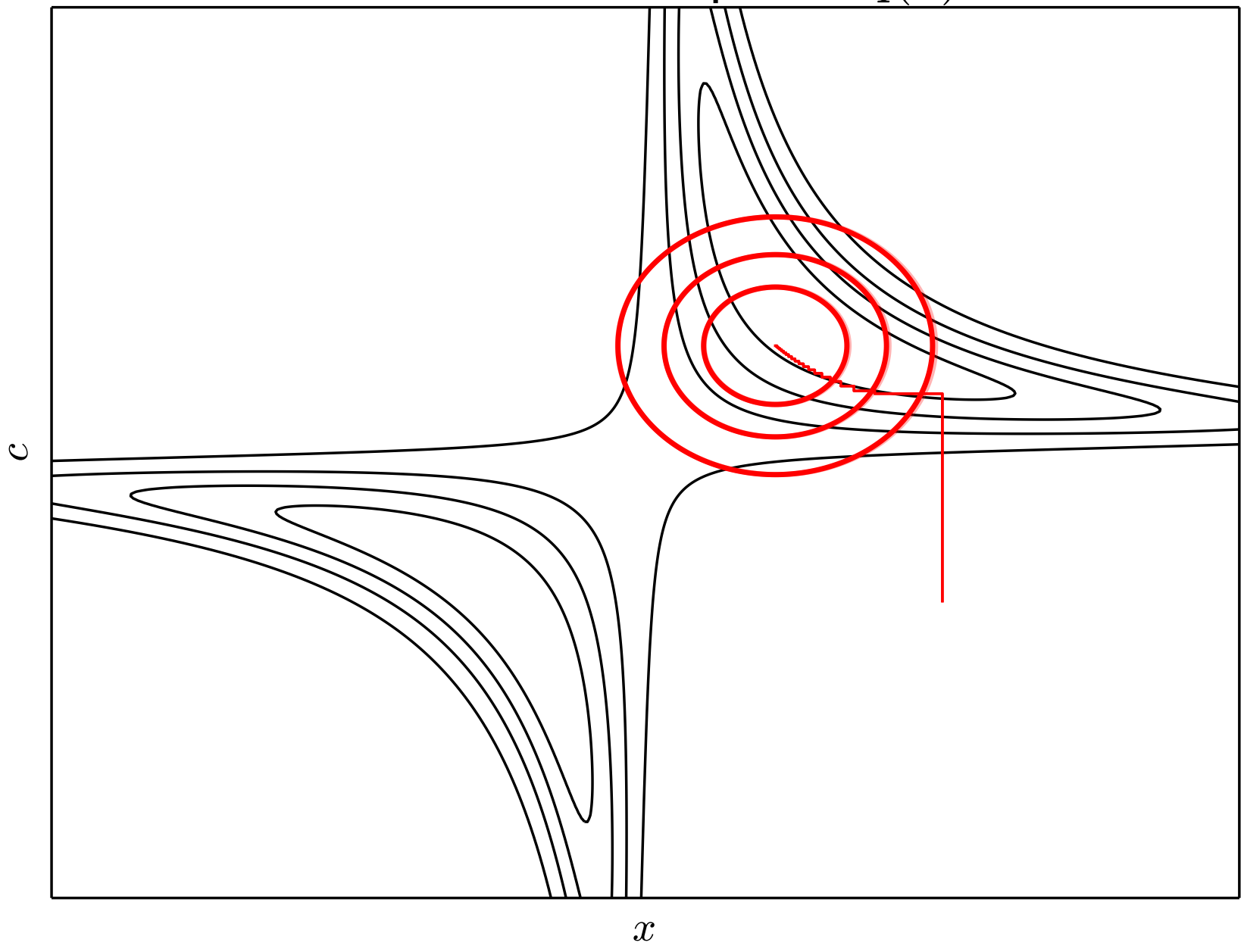
Iteration 18 - Update $q(x)$



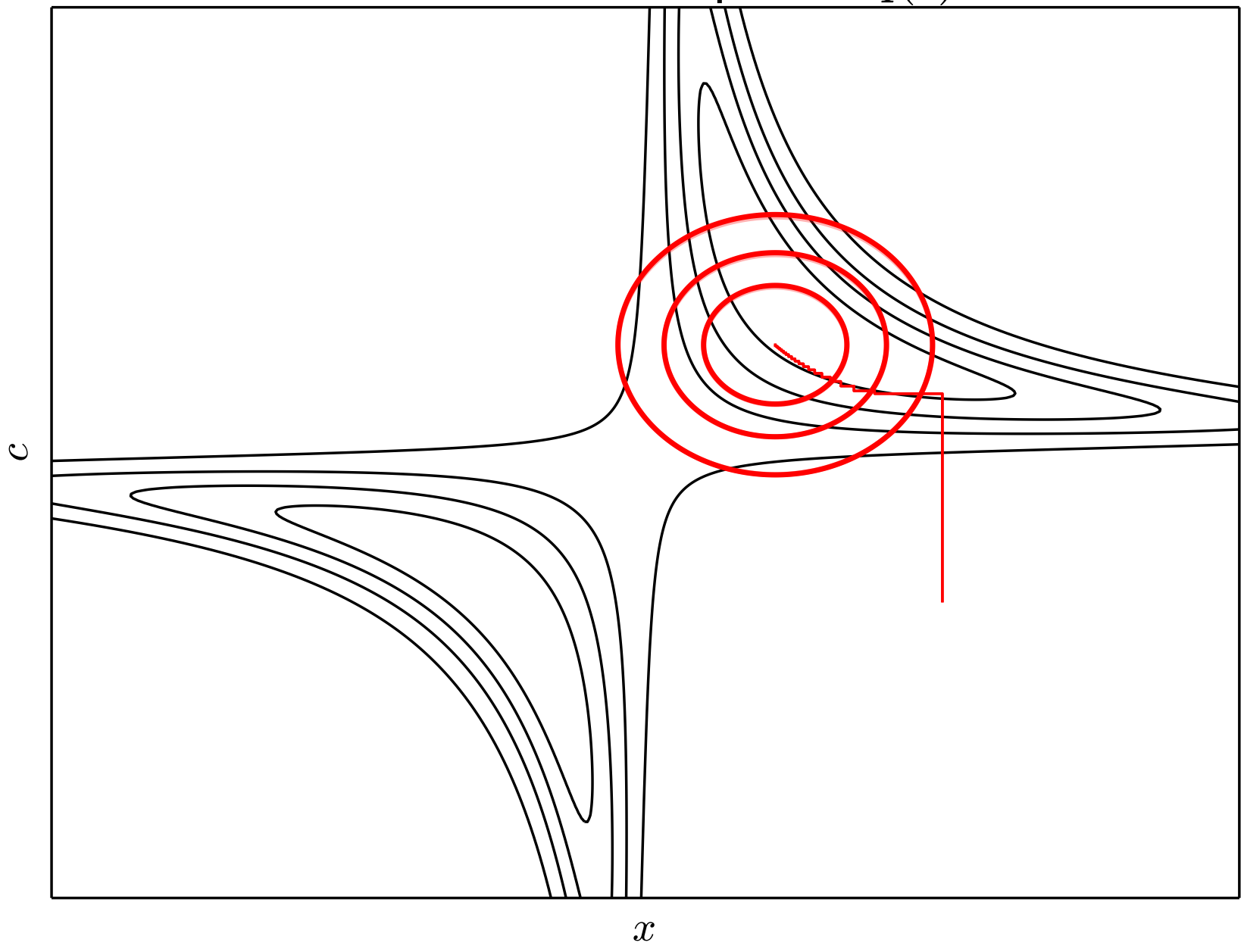
Iteration 19 - Update $q(c)$



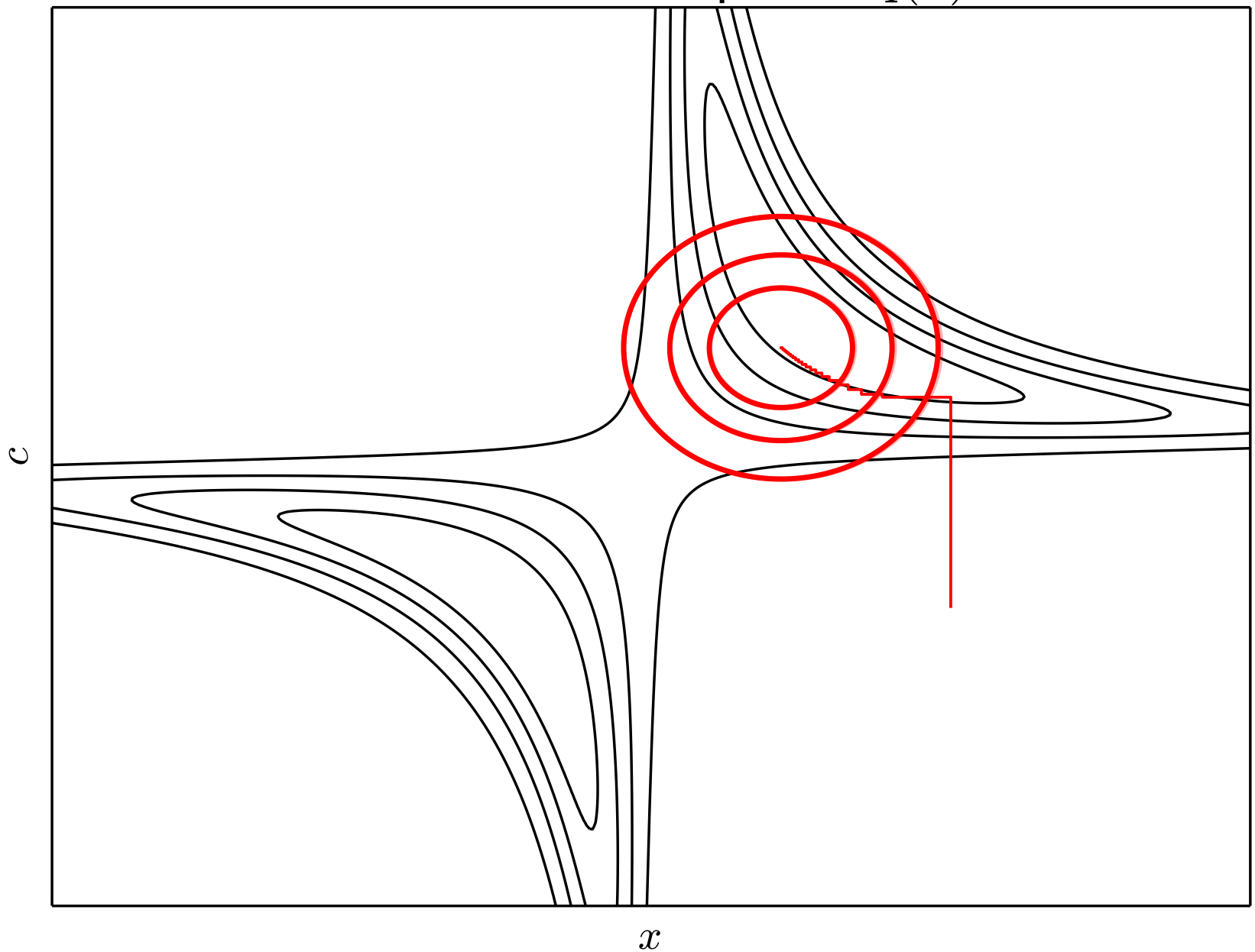
Iteration 19 - Update $q(x)$



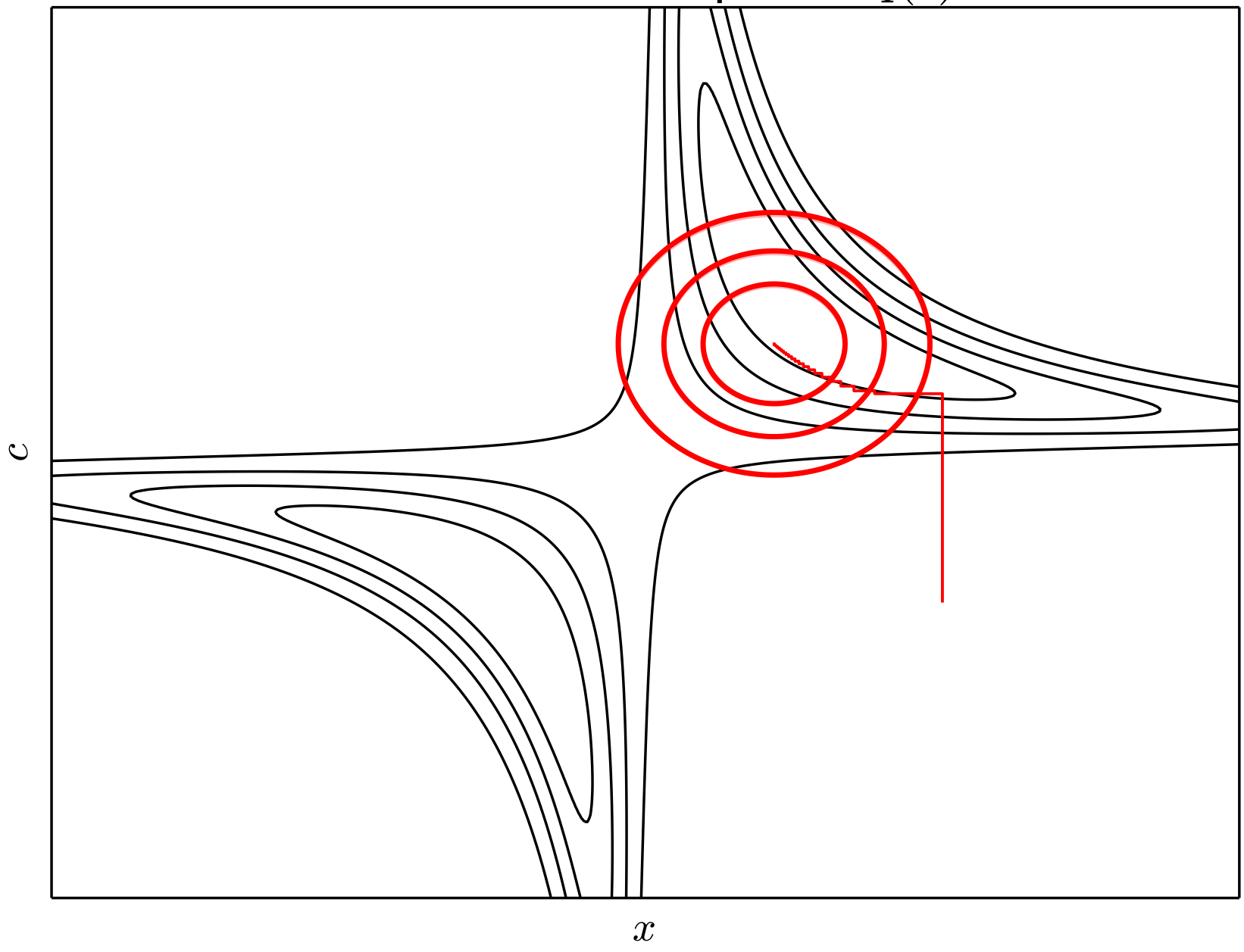
Iteration 20 - Update $q(c)$



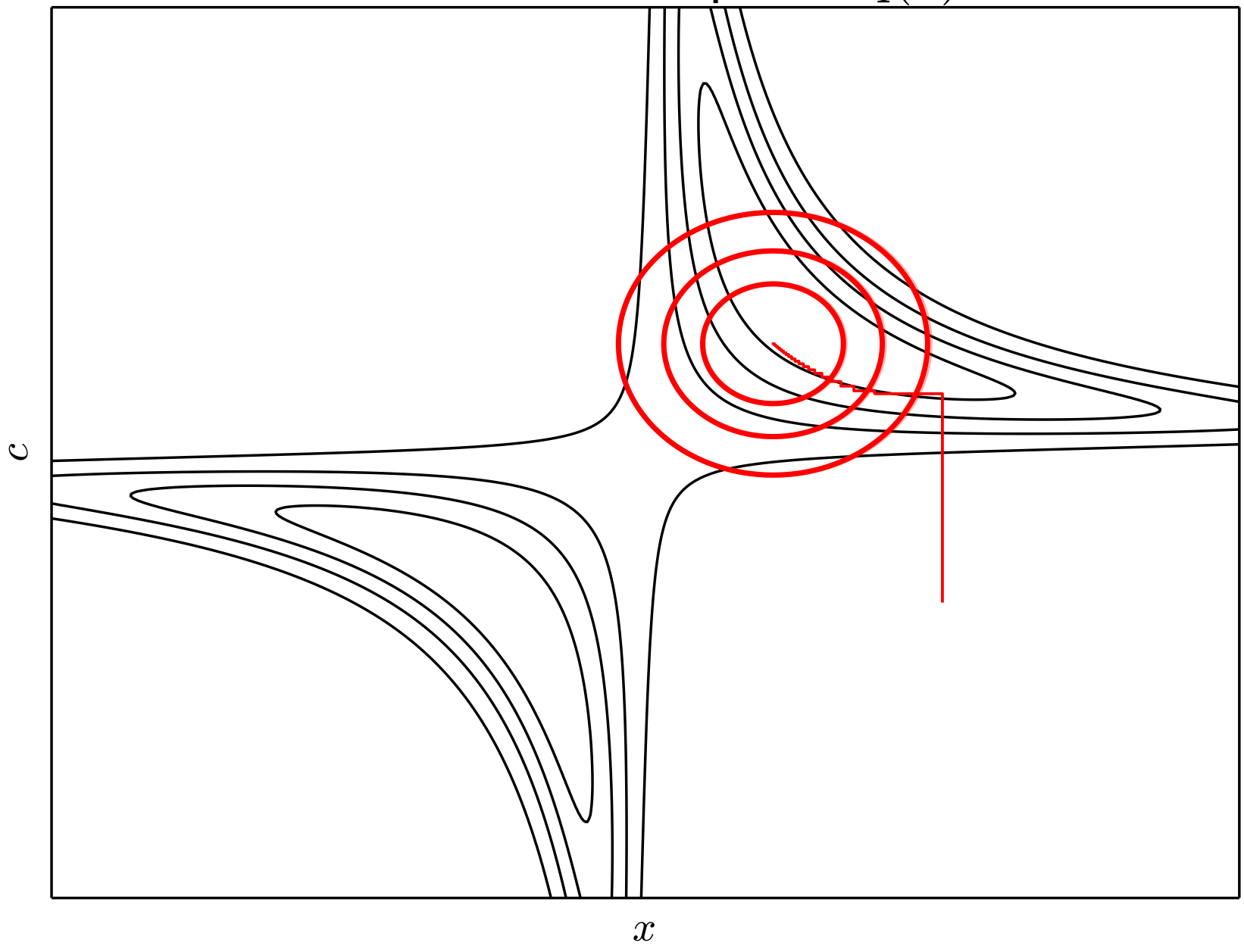
Iteration 20 - Update $q(x)$



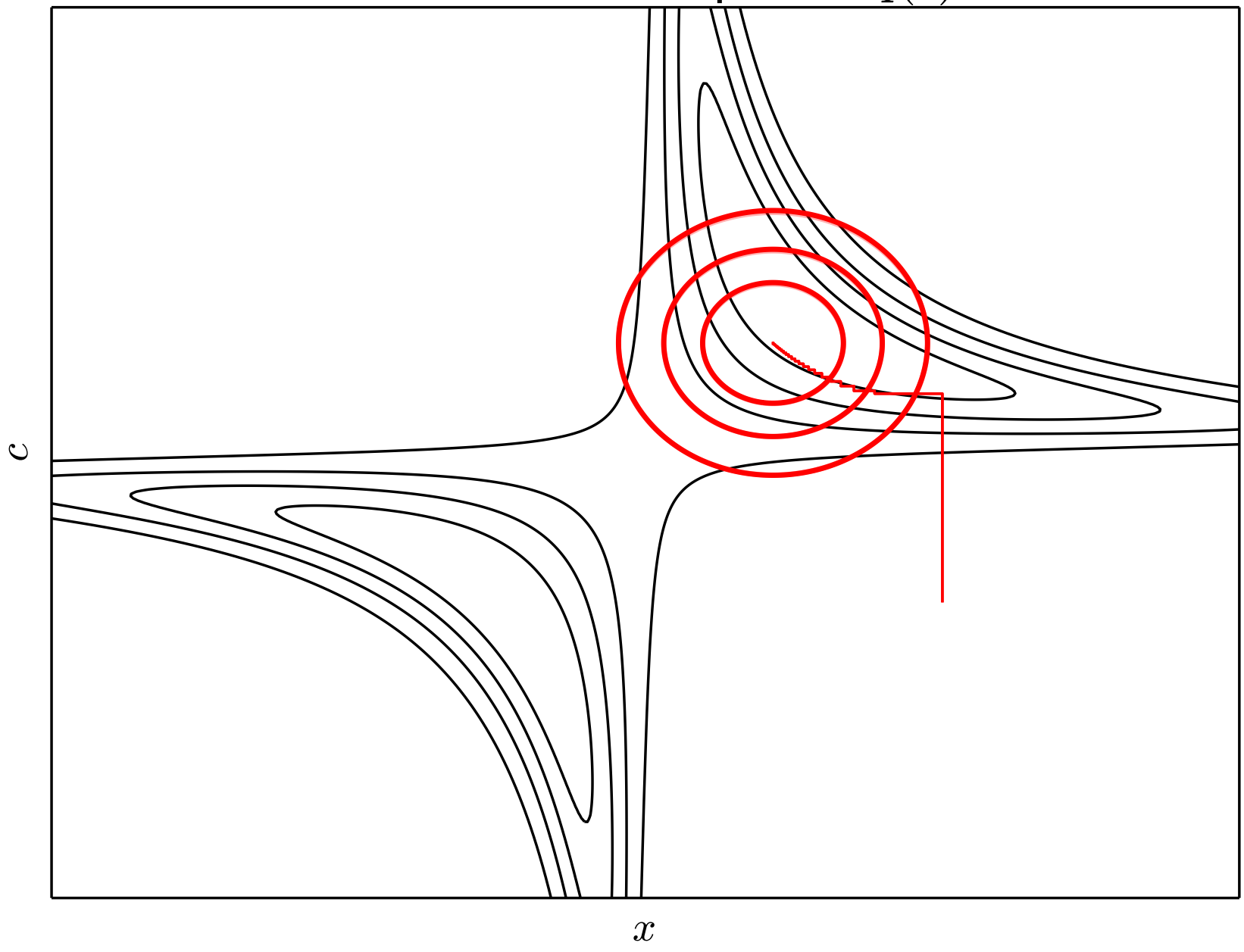
Iteration 21 - Update $q(c)$



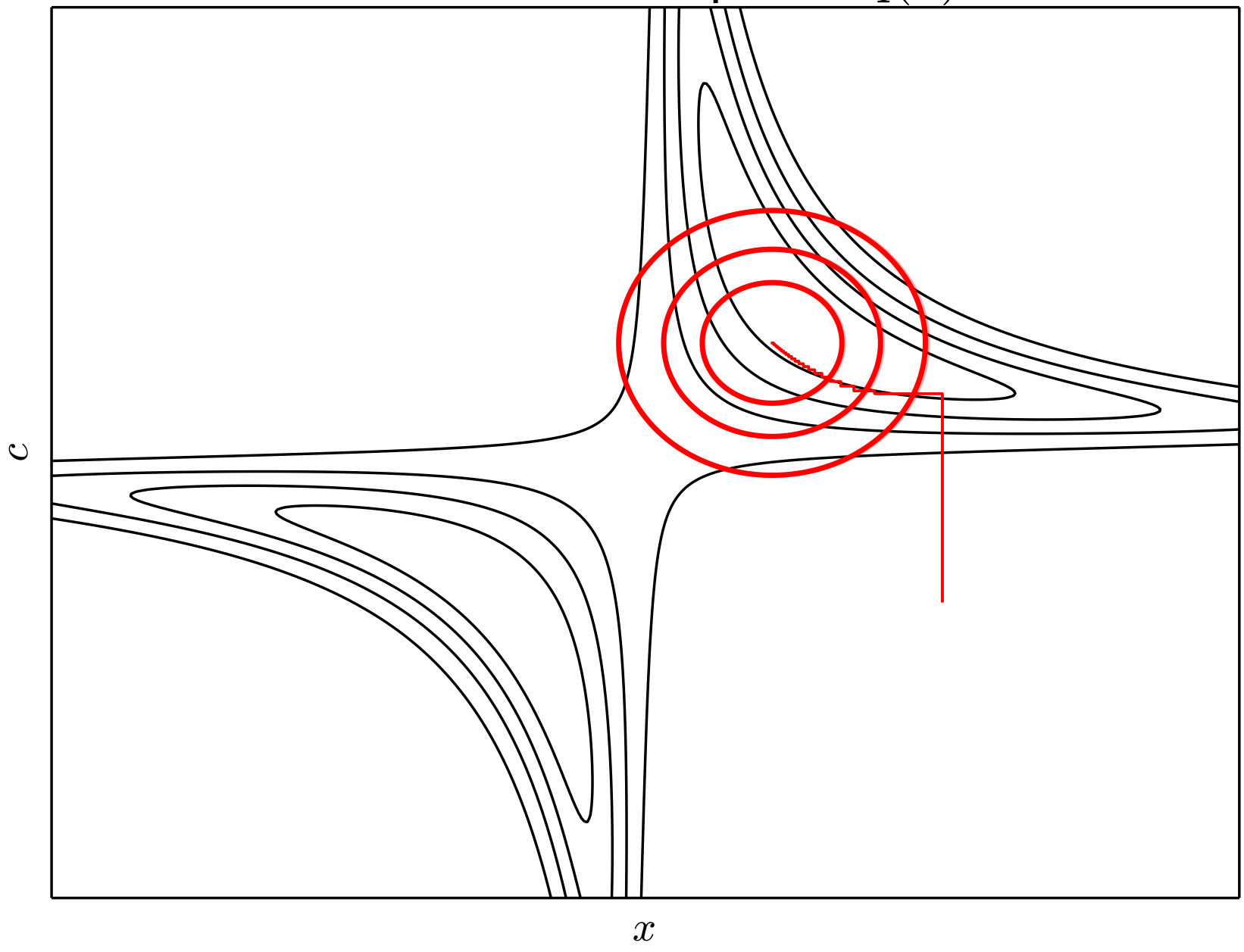
Iteration 21 - Update $q(x)$



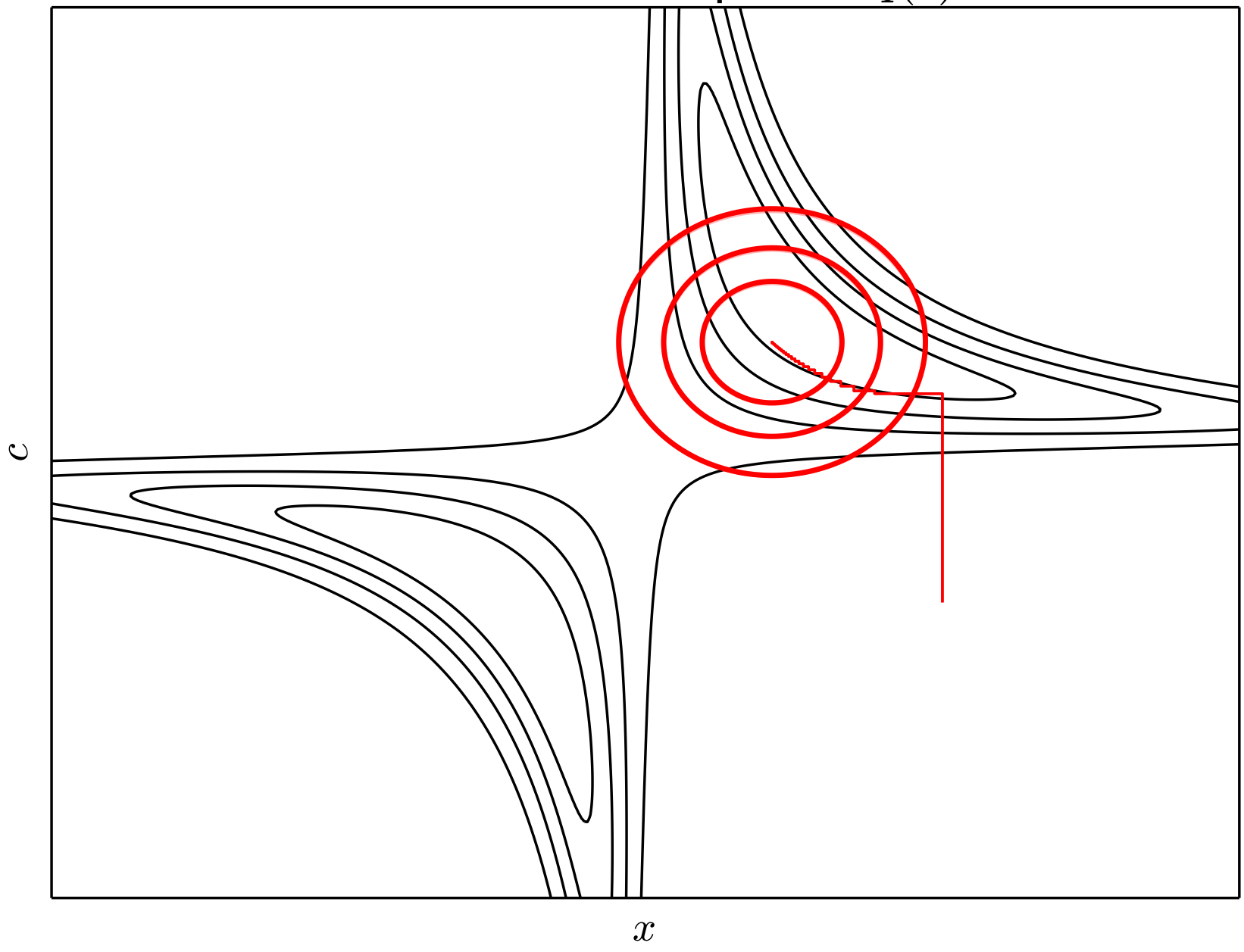
Iteration 22 - Update $q(c)$



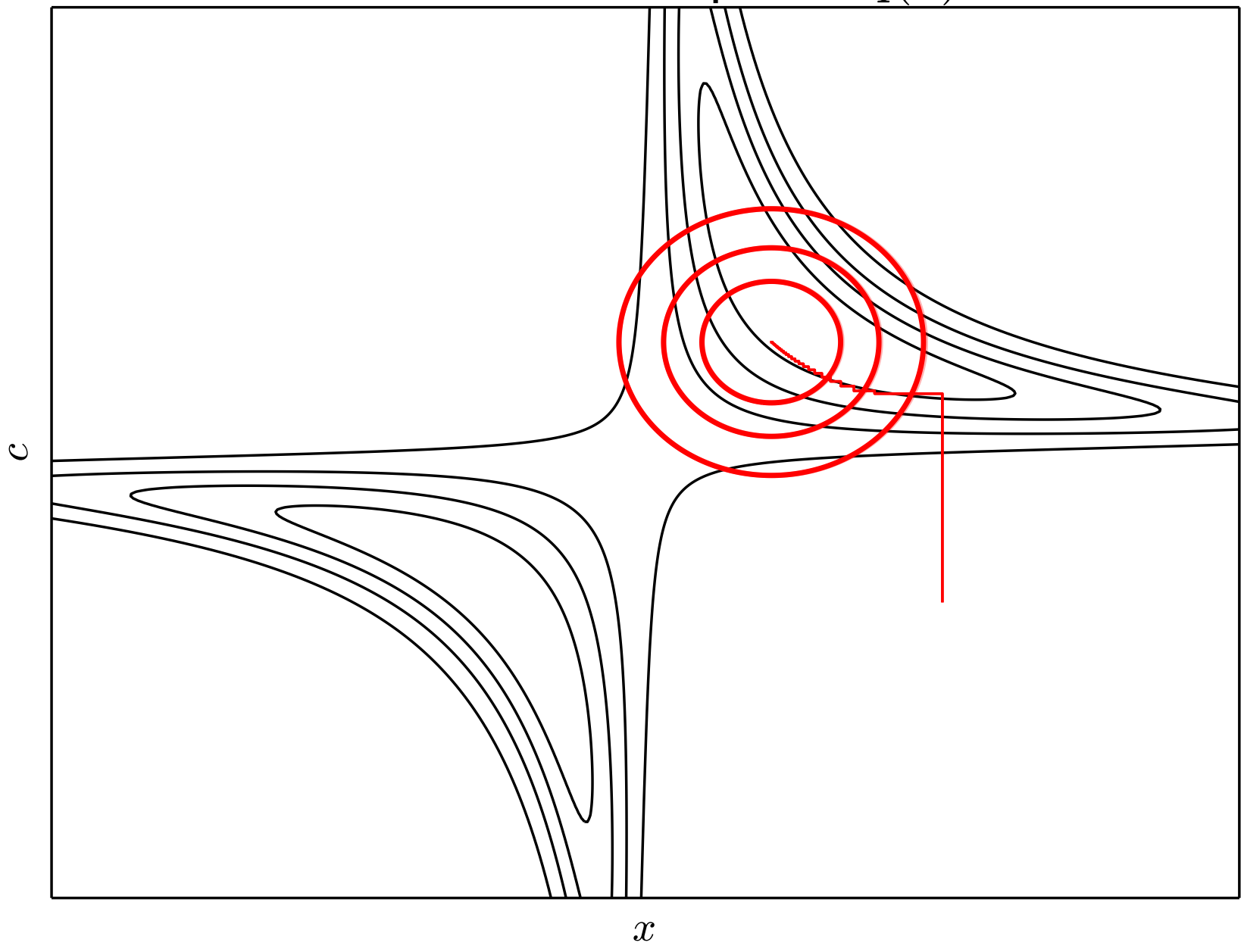
Iteration 22 - Update $q(x)$



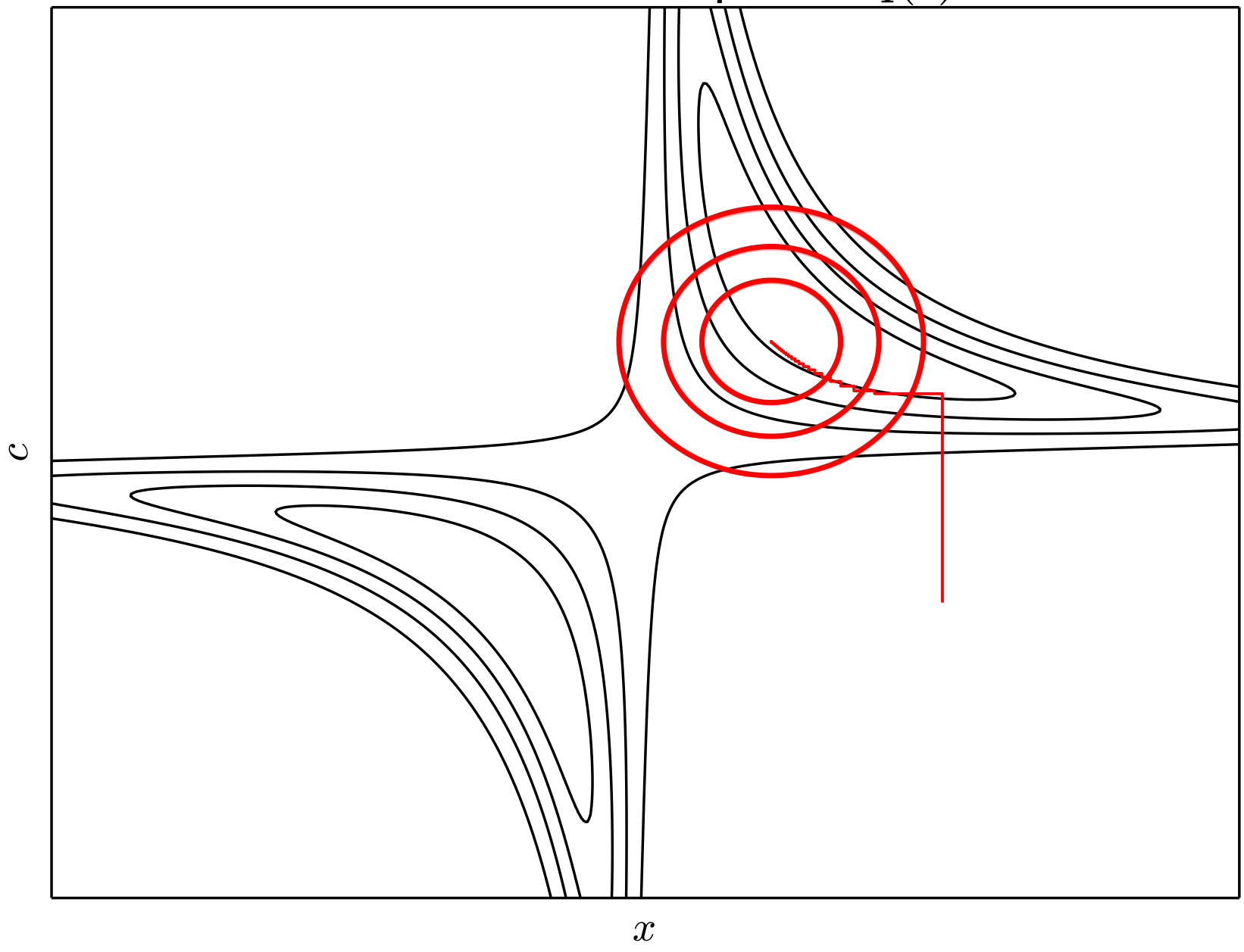
Iteration 23 - Update $q(c)$



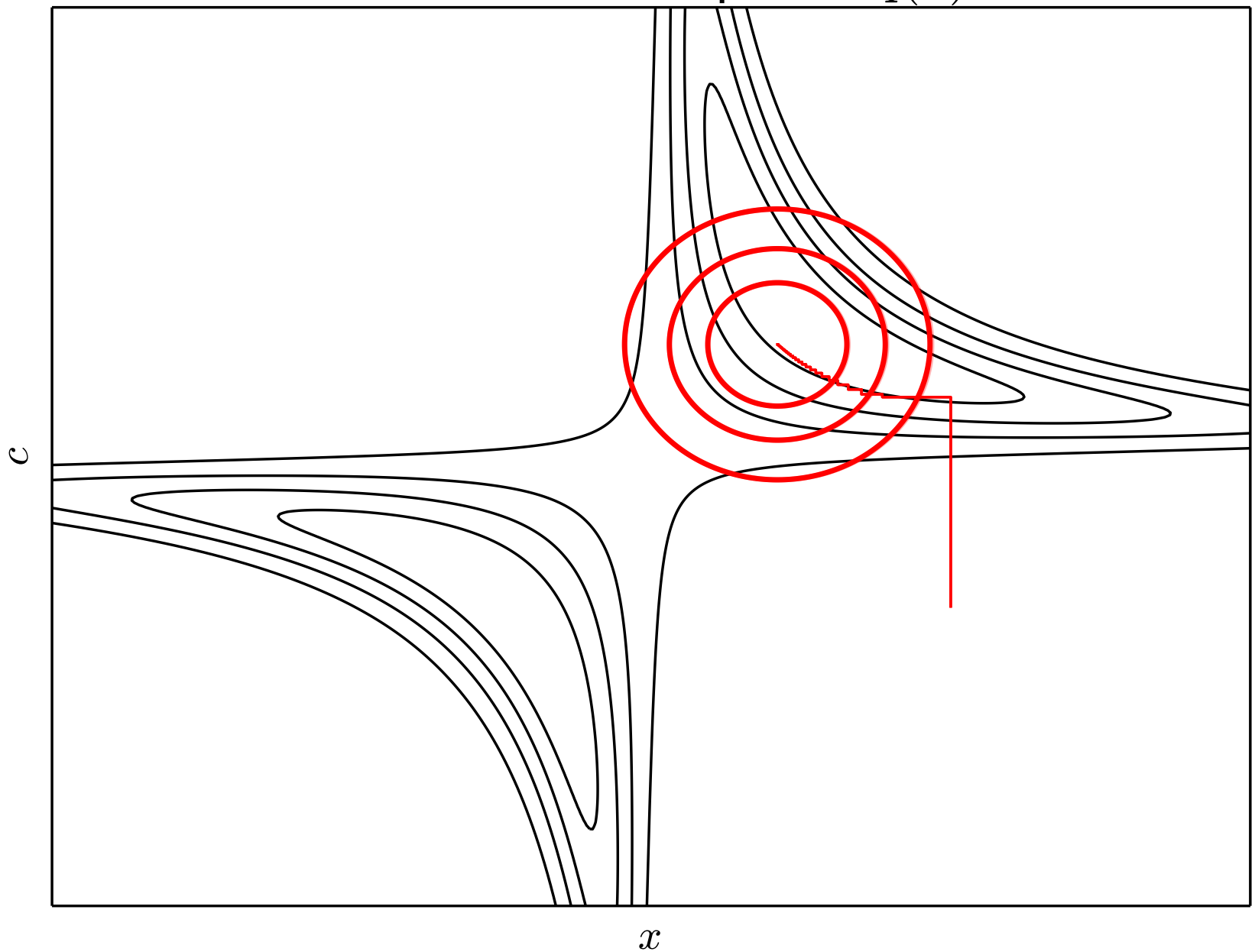
Iteration 23 - Update $q(x)$



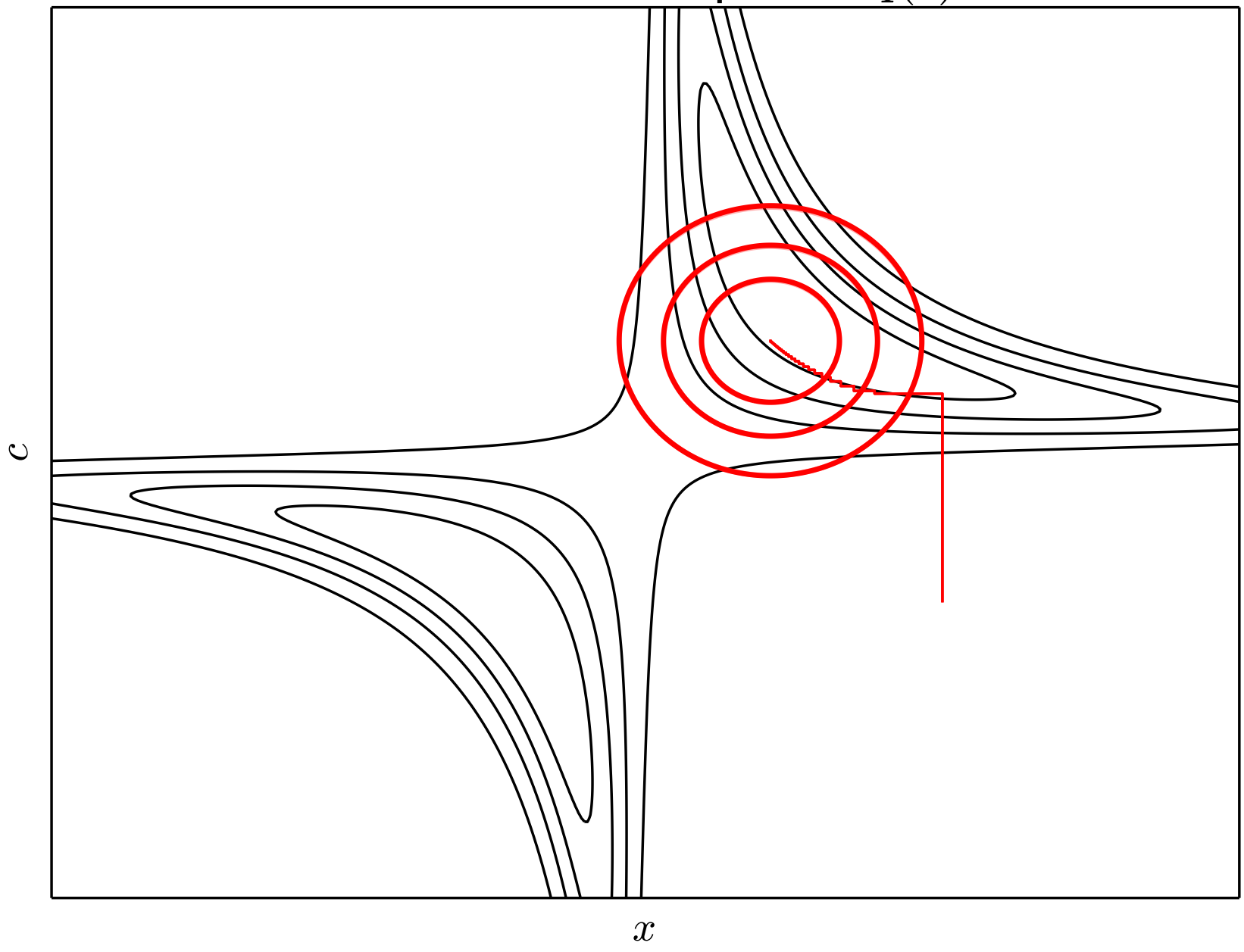
Iteration 24 - Update $q(c)$



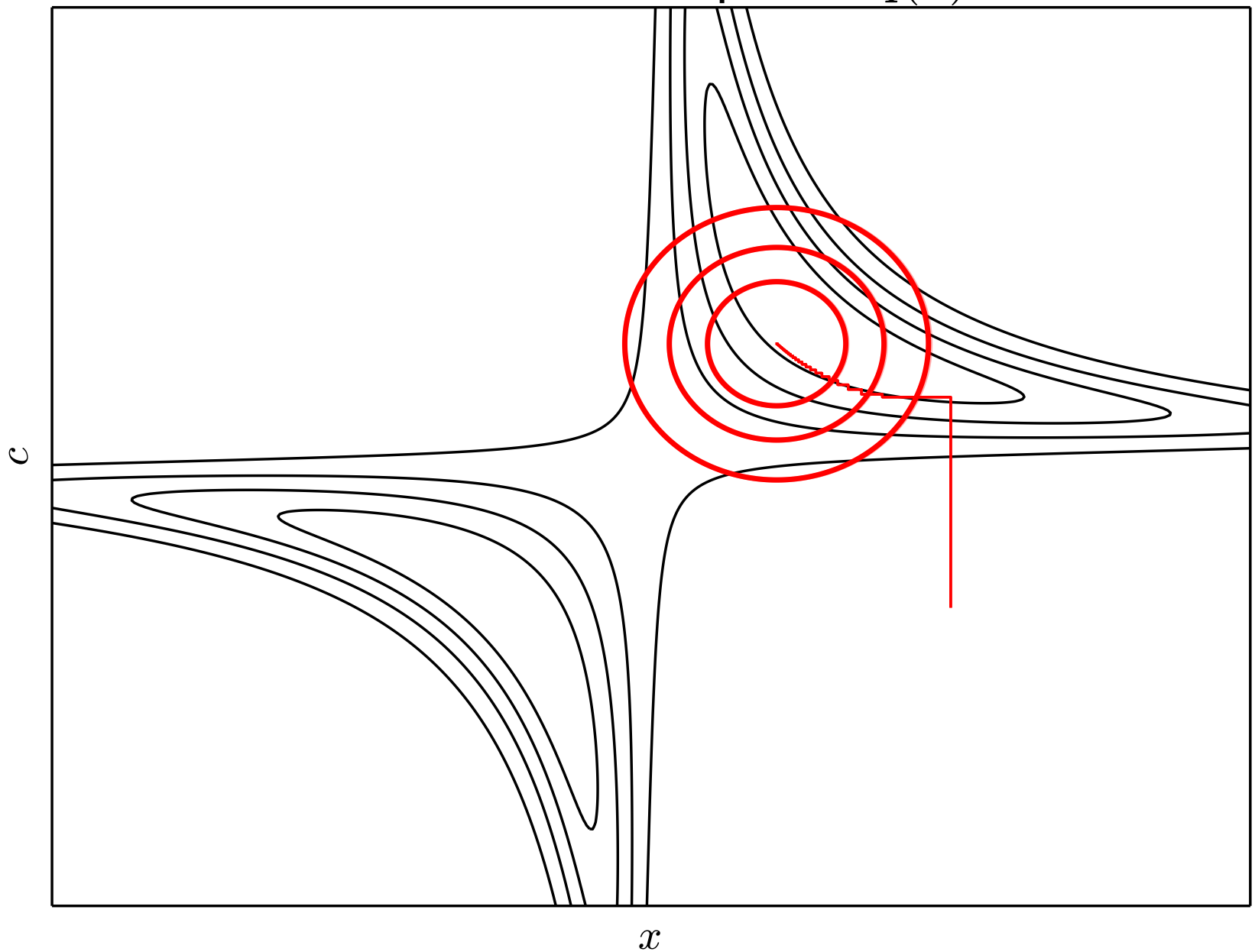
Iteration 24 - Update $q(x)$



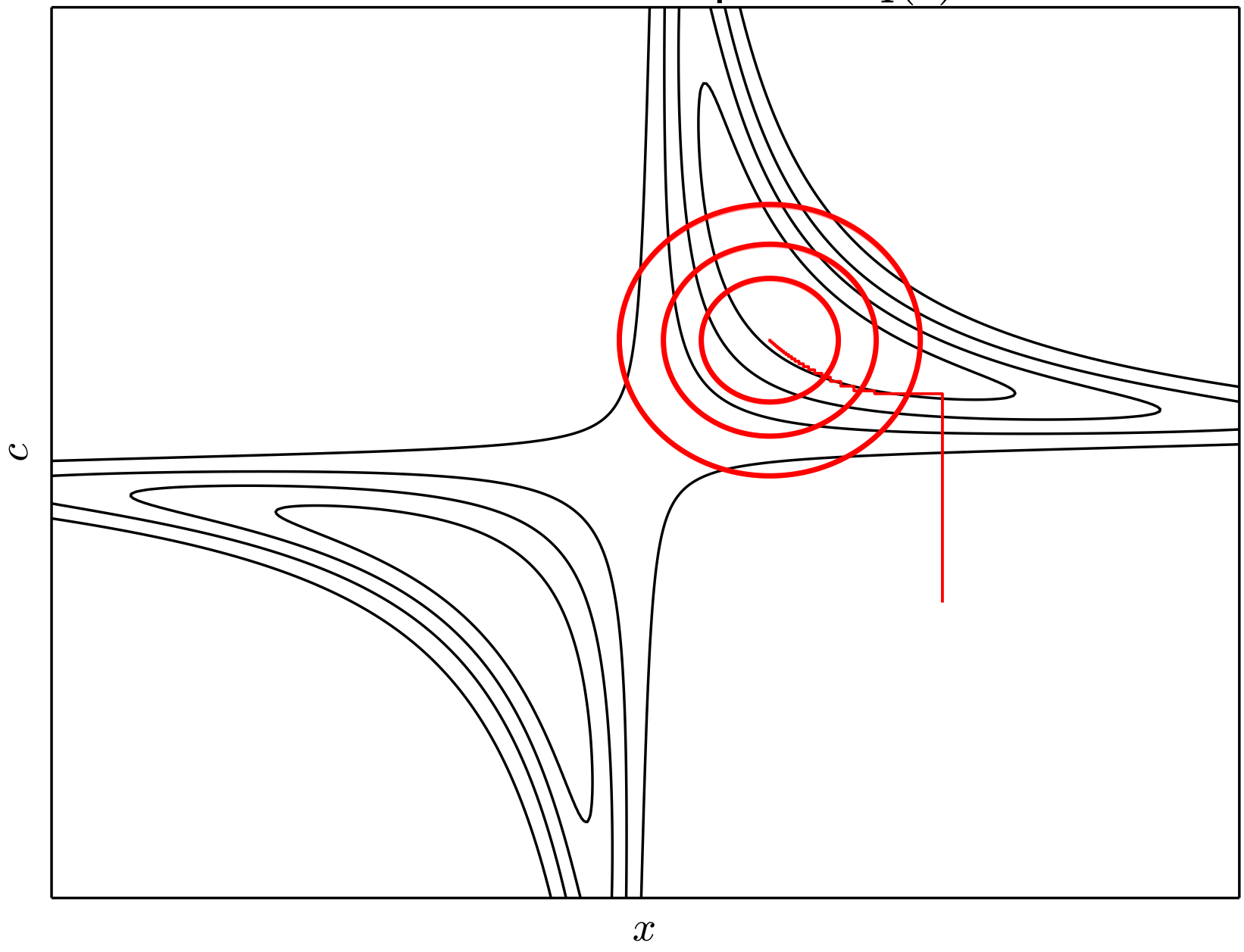
Iteration 25 - Update $q(c)$



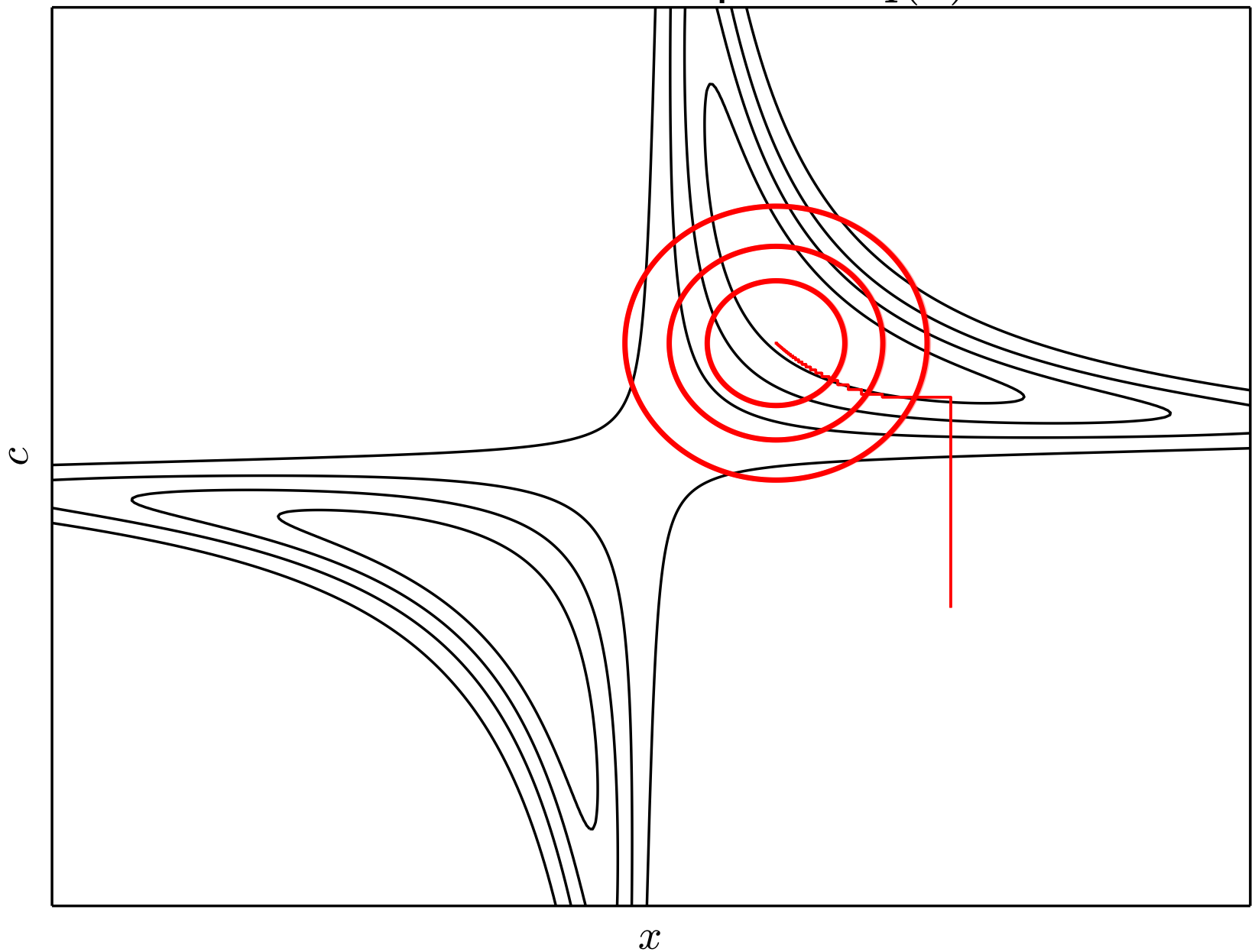
Iteration 25 - Update $q(x)$



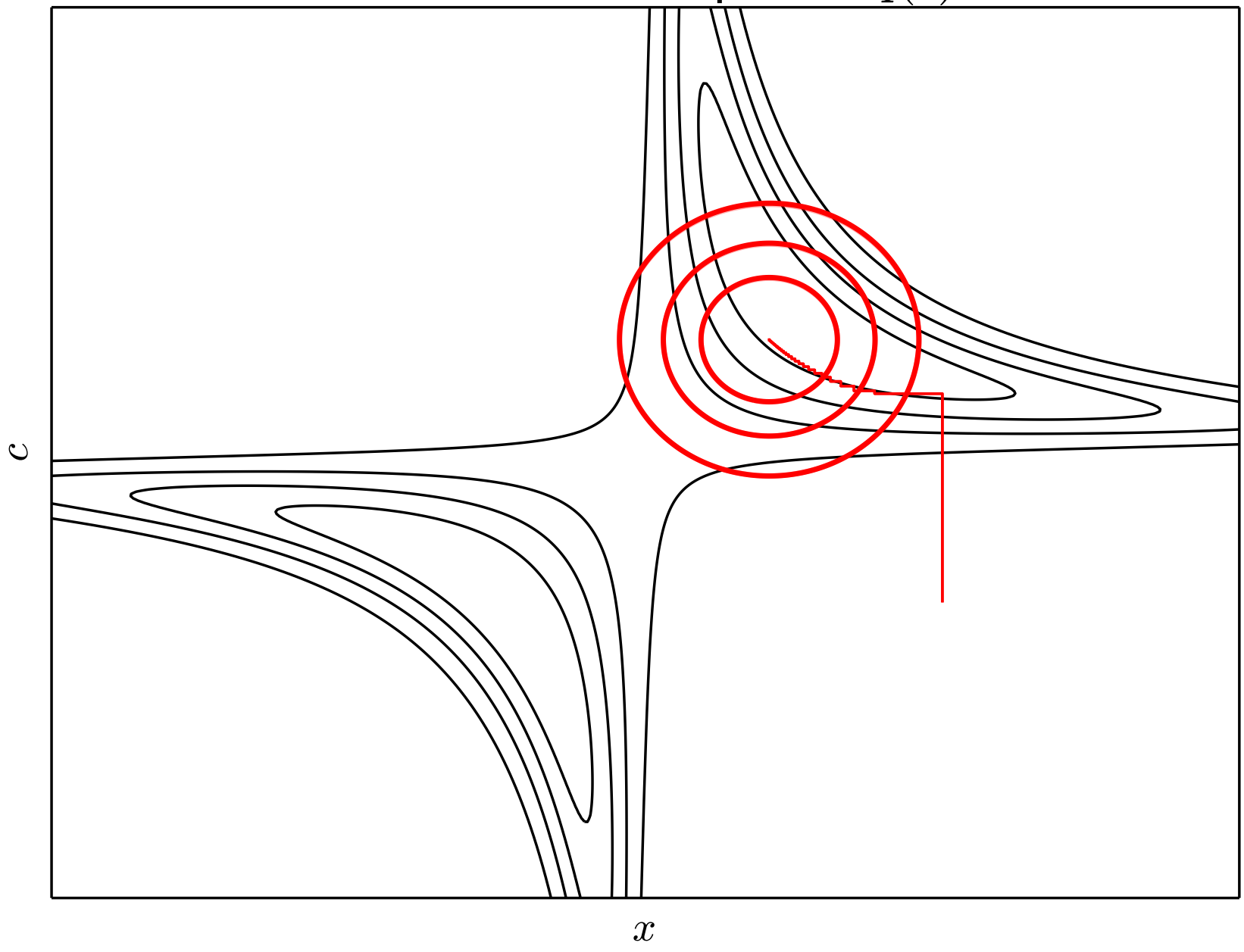
Iteration 26 - Update $q(c)$



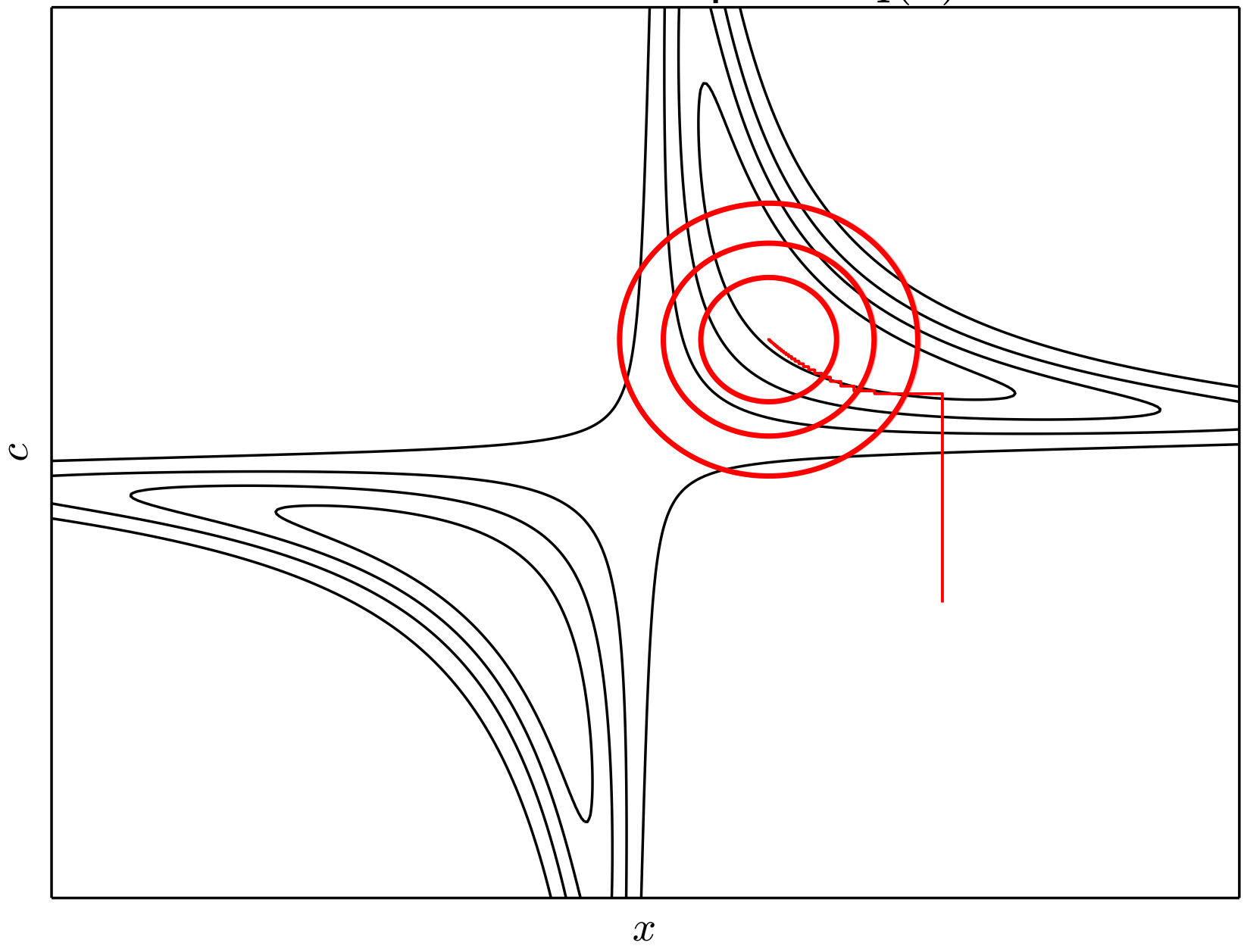
Iteration 26 - Update $q(x)$



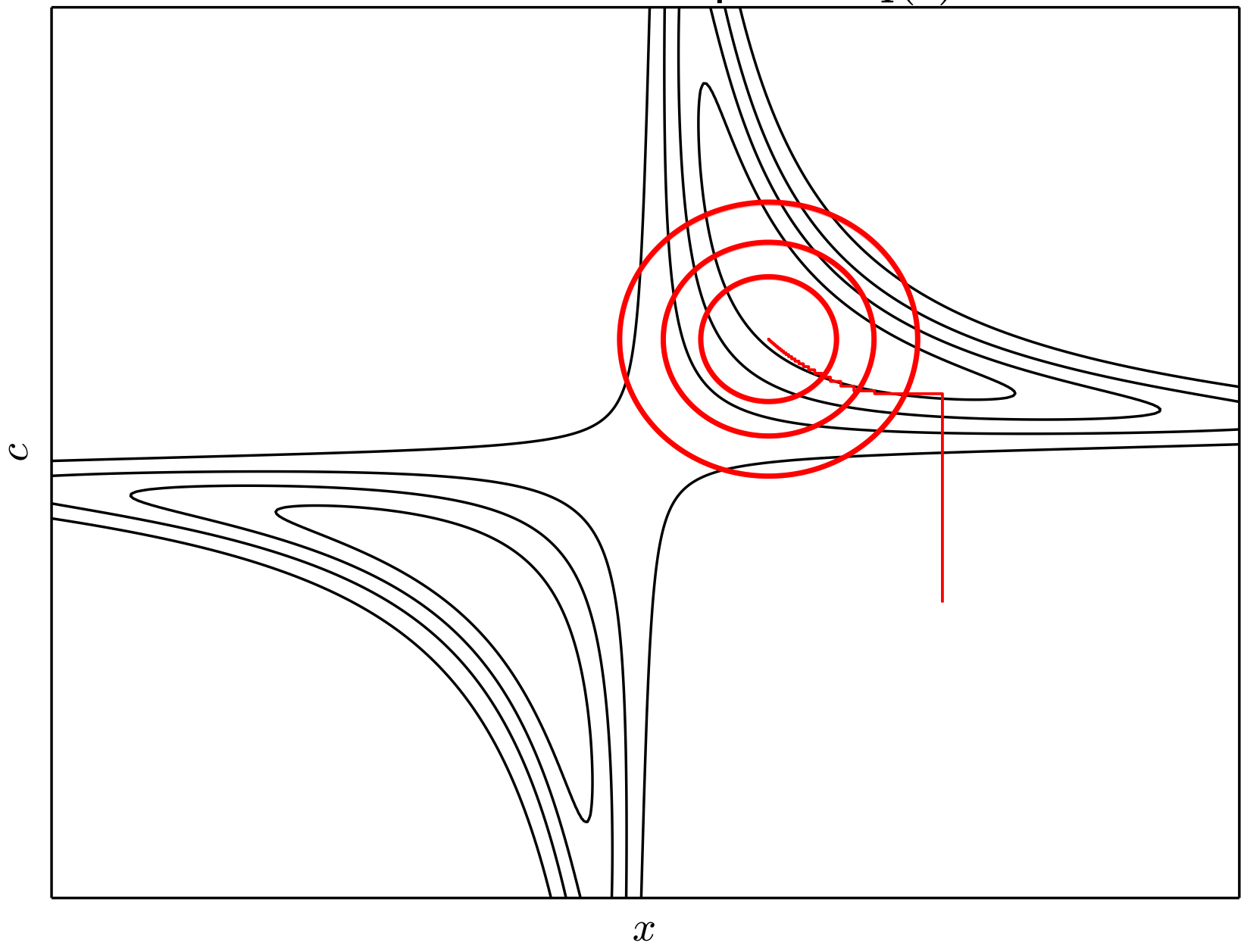
Iteration 27 - Update $q(c)$



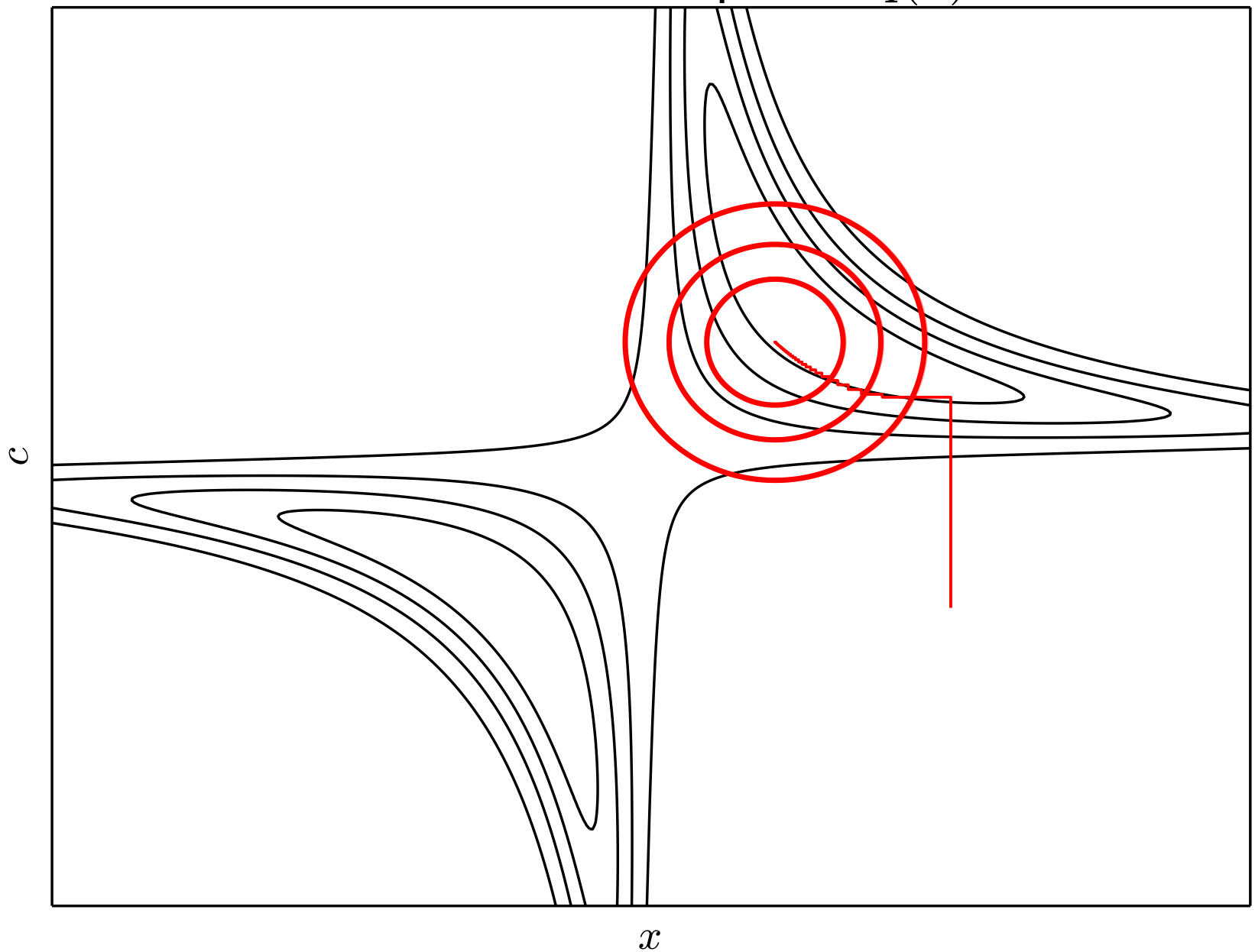
Iteration 27 - Update $q(x)$



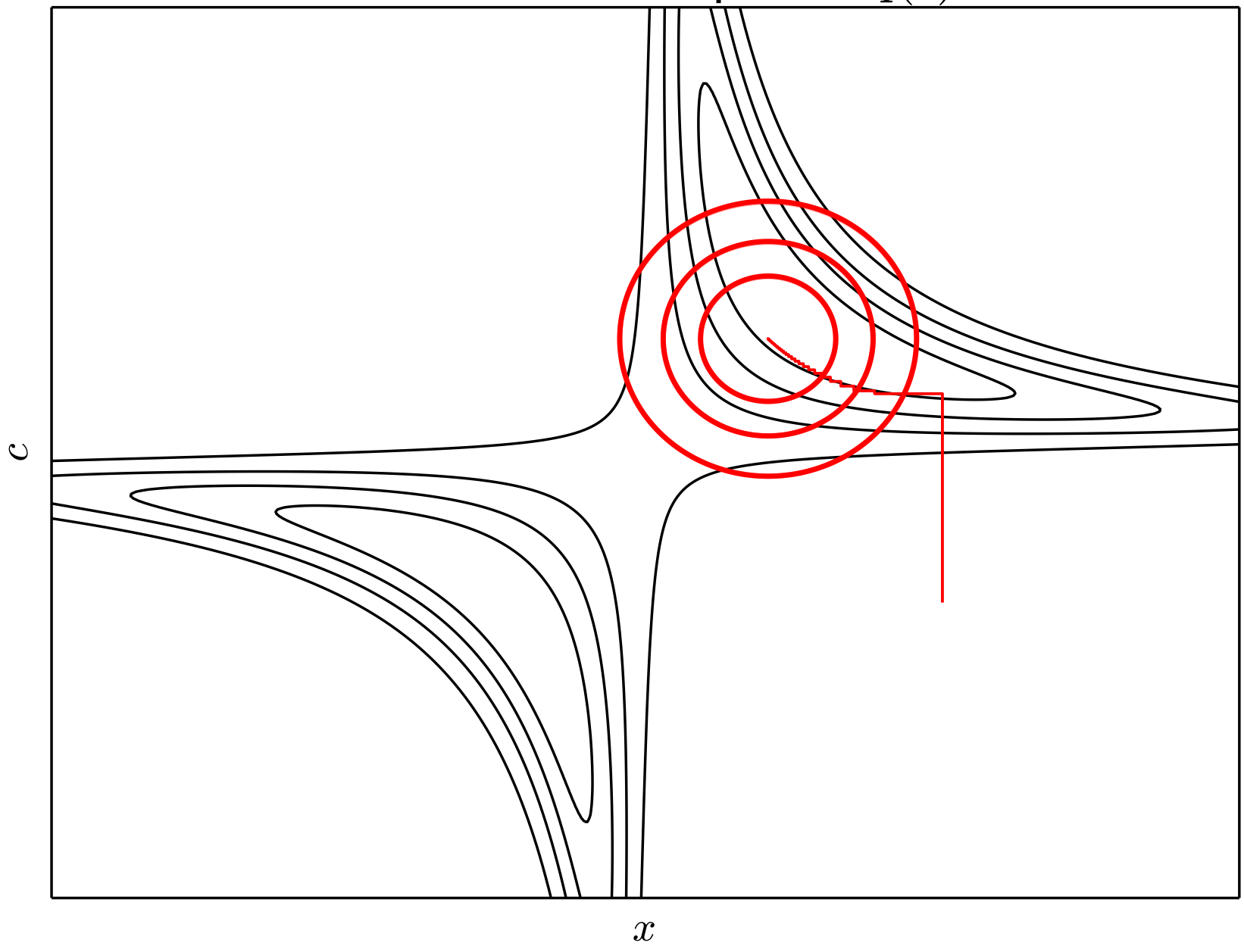
Iteration 28 - Update $q(c)$



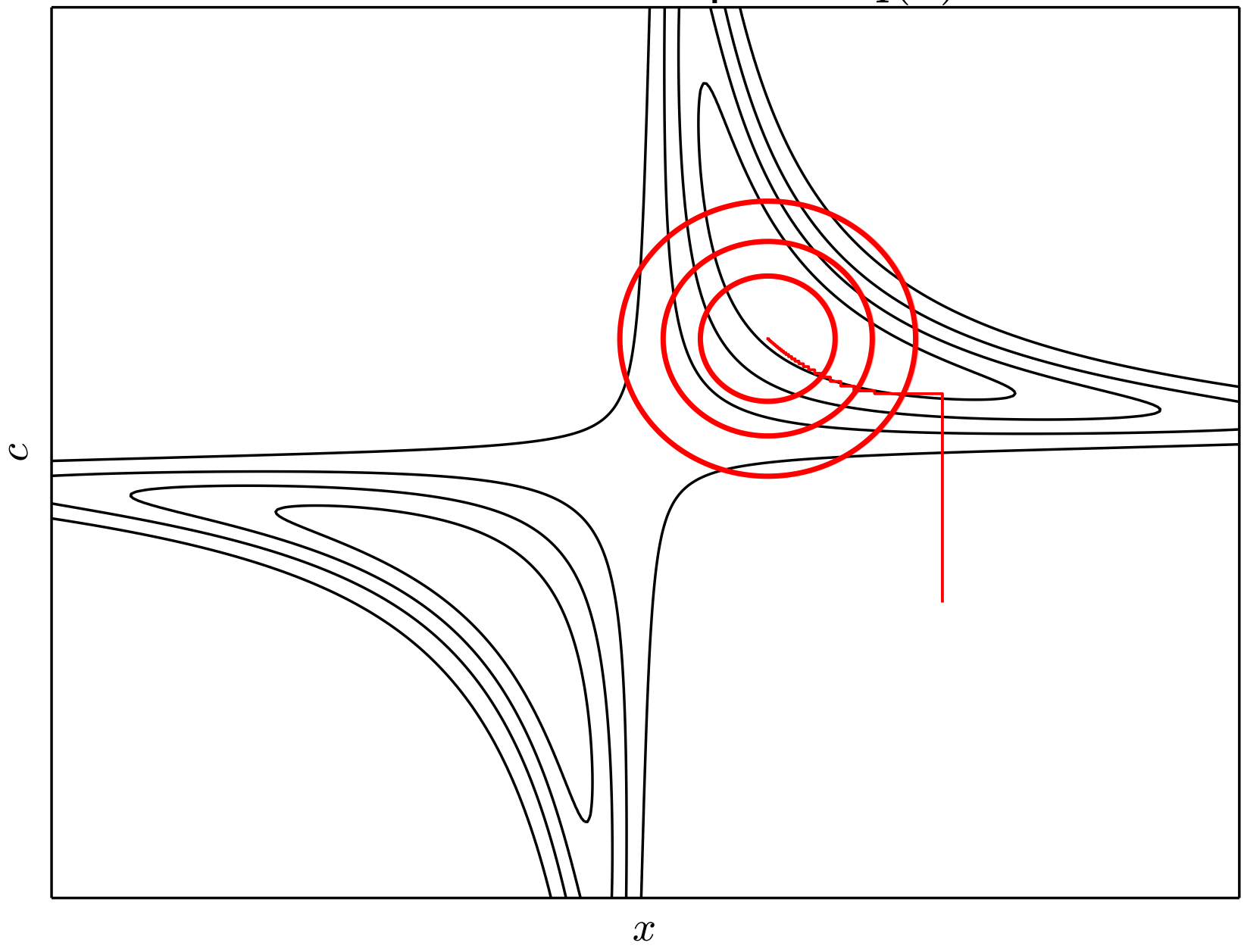
Iteration 28 - Update $q(x)$



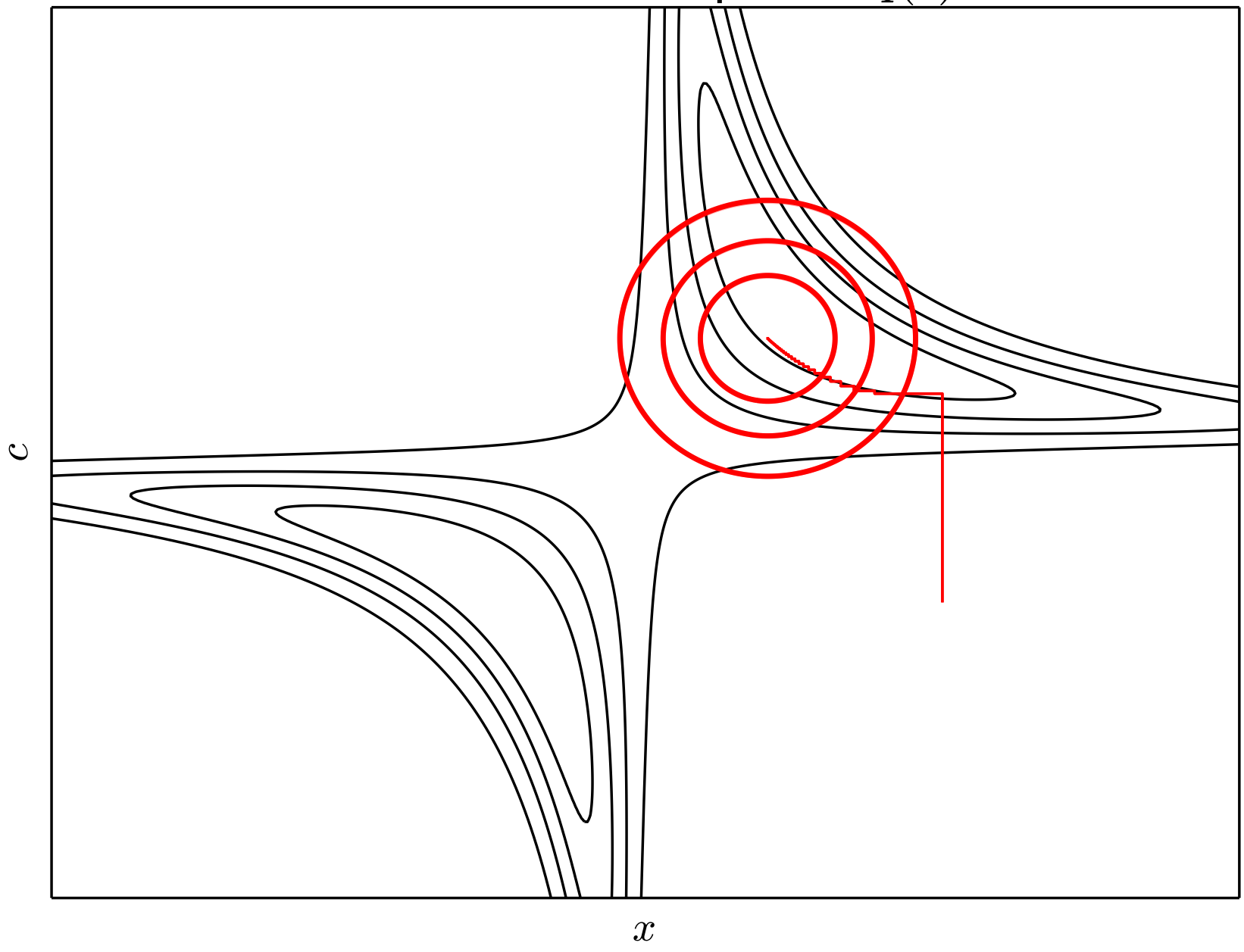
Iteration 29 - Update $q(c)$



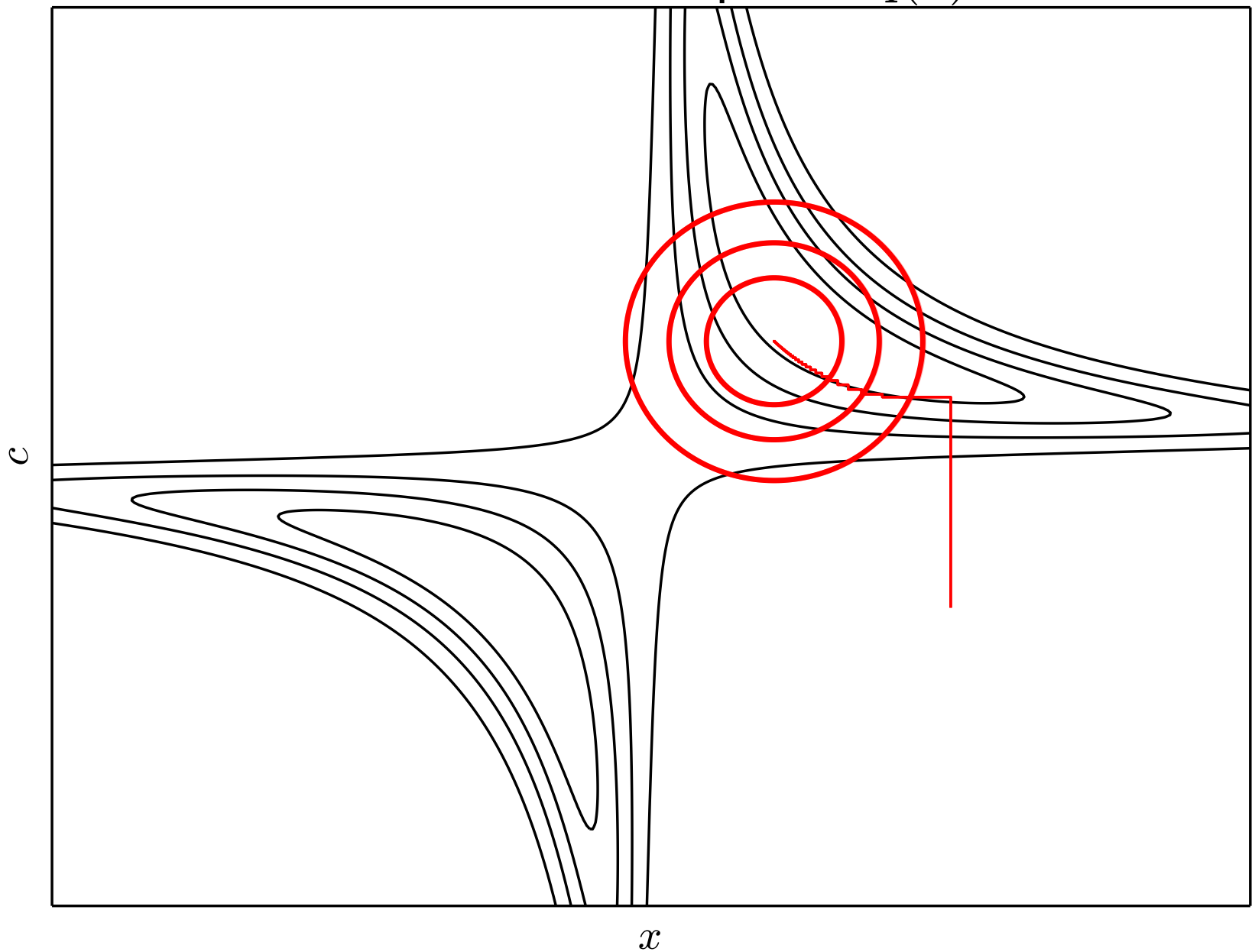
Iteration 29 - Update $q(x)$



Iteration 30 - Update $q(c)$



Iteration 30 - Update $q(x)$



Problem: Slow convergence

- How to avoid zigzagging?

- Does the convergence matter in practice?

Solution: Parameter expansion

- Find a parameterization related to the coupling
- Joint optimize the variables using the parameterization
- Known as parameter expansion

Solution: Parameter expansion

- The states can be rotated by compensating it in the loadings:

$$\mathbf{y}_n = \mathbf{C}\mathbf{x}_n = \mathbf{C}\mathbf{R}^{-1}\mathbf{R}\mathbf{x}_n \,,$$

thus rotate as $\mathbf{C} \to \mathbf{C}\mathbf{R}^{-1}$ and $\mathbf{x}_n \to \mathbf{R}\mathbf{x}_n$.

Solution: Parameter expansion

- The states can be rotated by compensating it in the loadings:

$$\mathbf{y}_n = \mathbf{C}\mathbf{x}_n = \mathbf{C}\mathbf{R}^{-1}\mathbf{R}\mathbf{x}_n \,,$$

  thus rotate as $\mathbf{C} \to \mathbf{C}\mathbf{R}^{-1}$ and $\mathbf{x}_n \to \mathbf{R}\mathbf{x}_n$.

- Keep the dynamics of the latent states unaffected:

$$\mathbf{R}\mathbf{x}_n = \mathbf{R}\mathbf{A}\mathbf{R}^{-1}\mathbf{R}\mathbf{x}_{n-1} \,,$$

  thus rotate as $\mathbf{A} \to \mathbf{R}\mathbf{A}\mathbf{R}^{-1}$.

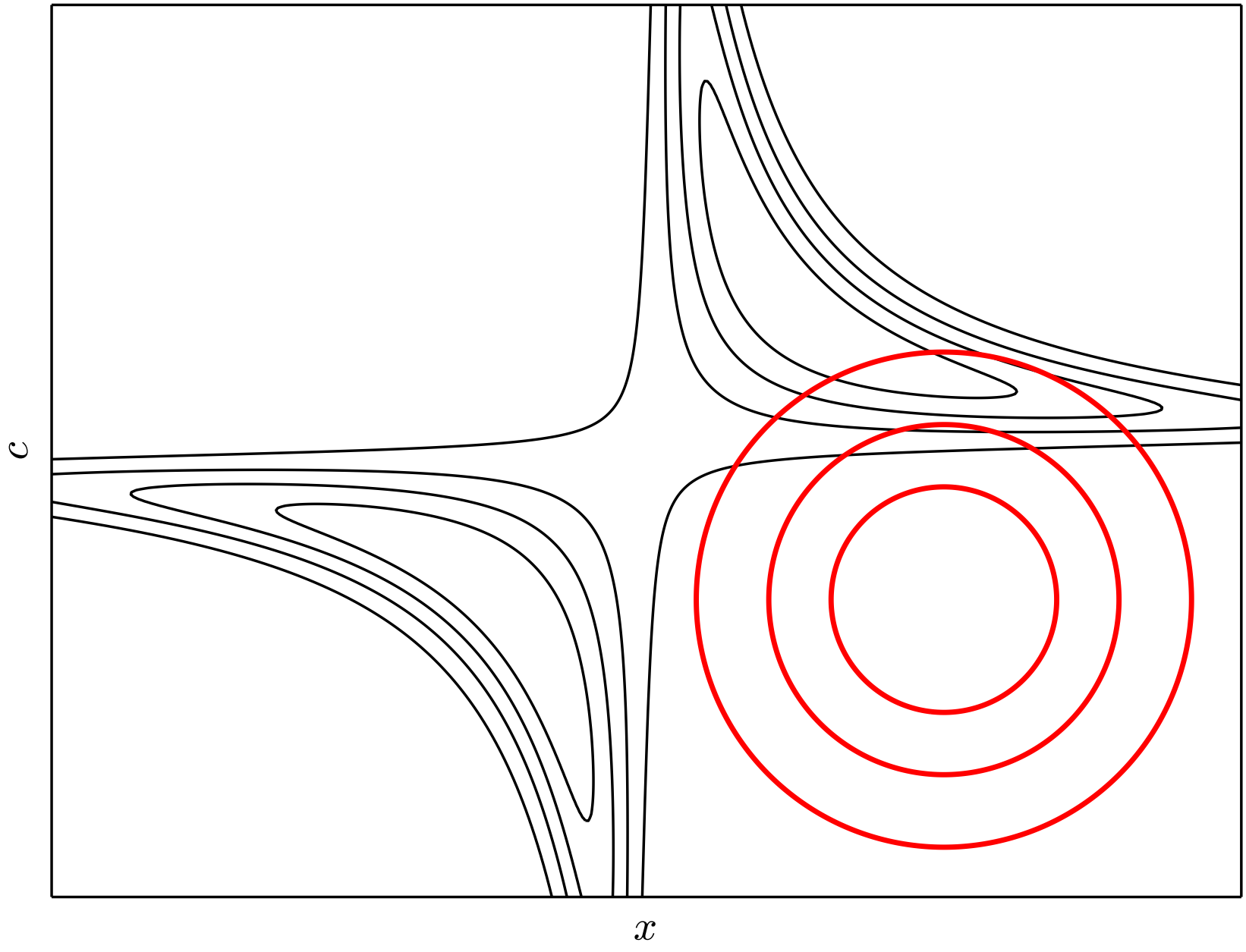
Simple demonstration

- 1-dimensional model $y = cx + \text{noise}$
- Approximate VB posterior:

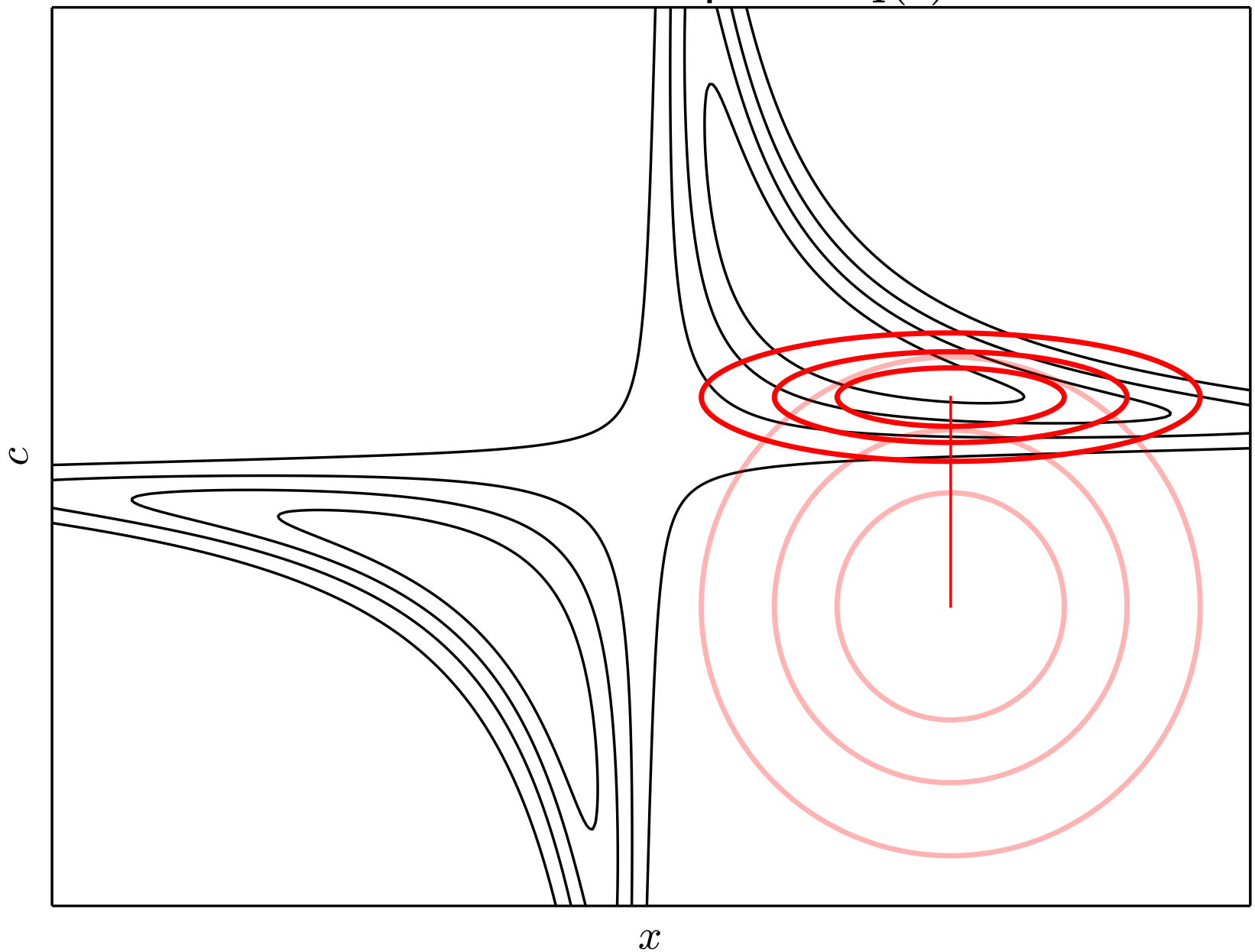
$$p(c, x|y) \approx q(c) \cdot q(x)$$

- Rotate as $x \to Rx$ and $c \to c/R$

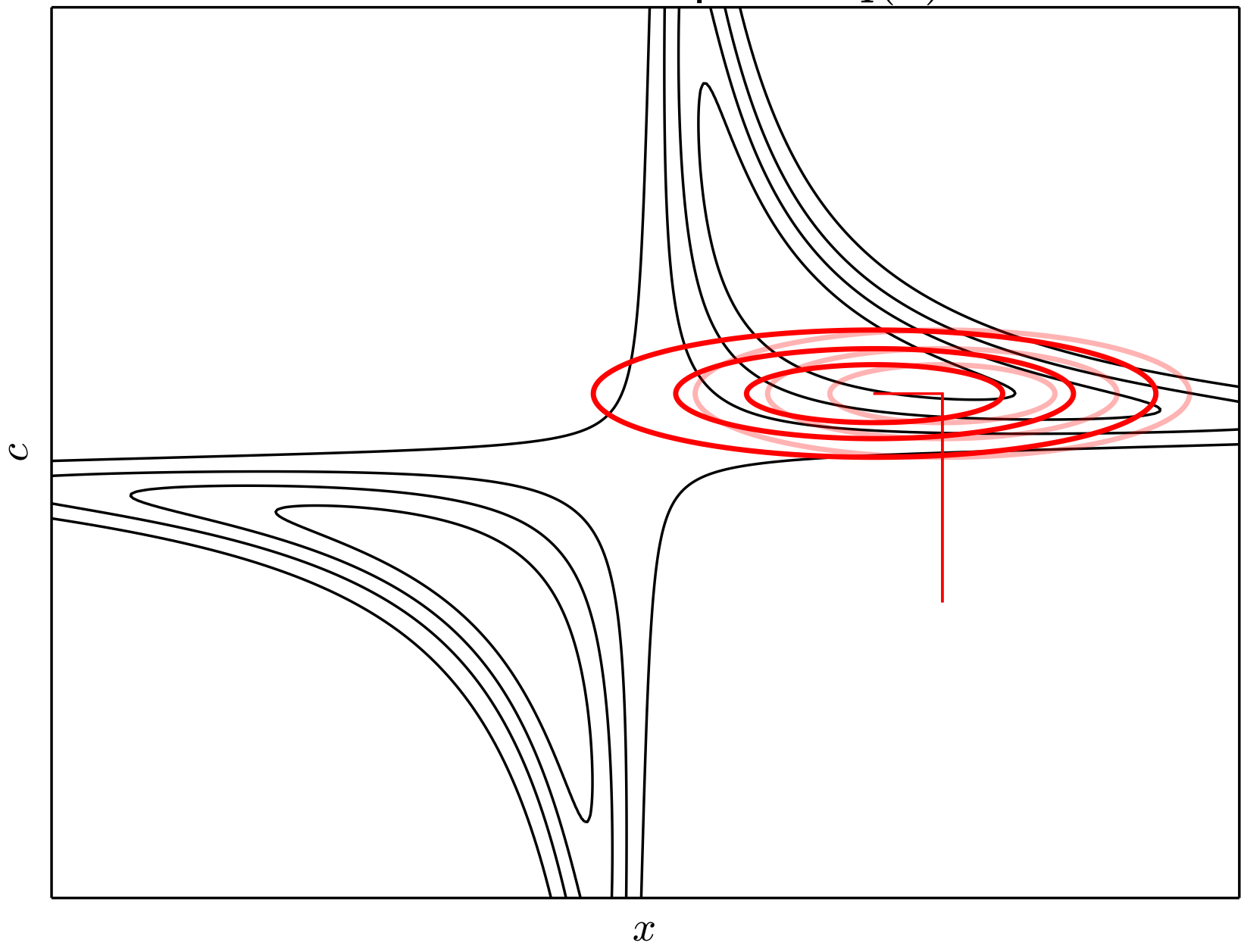
Initialization



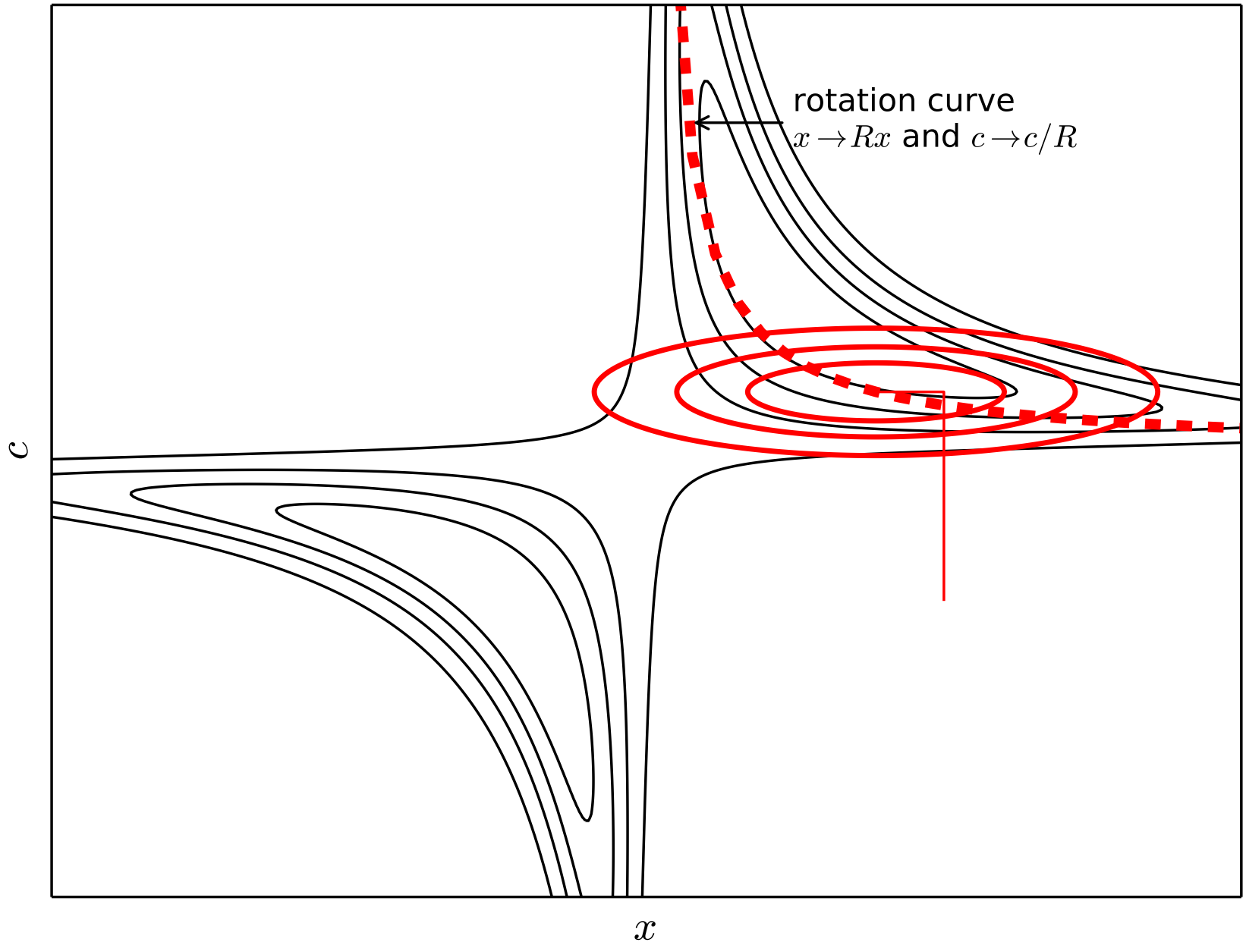
Iteration 1 - Update $q(c)$



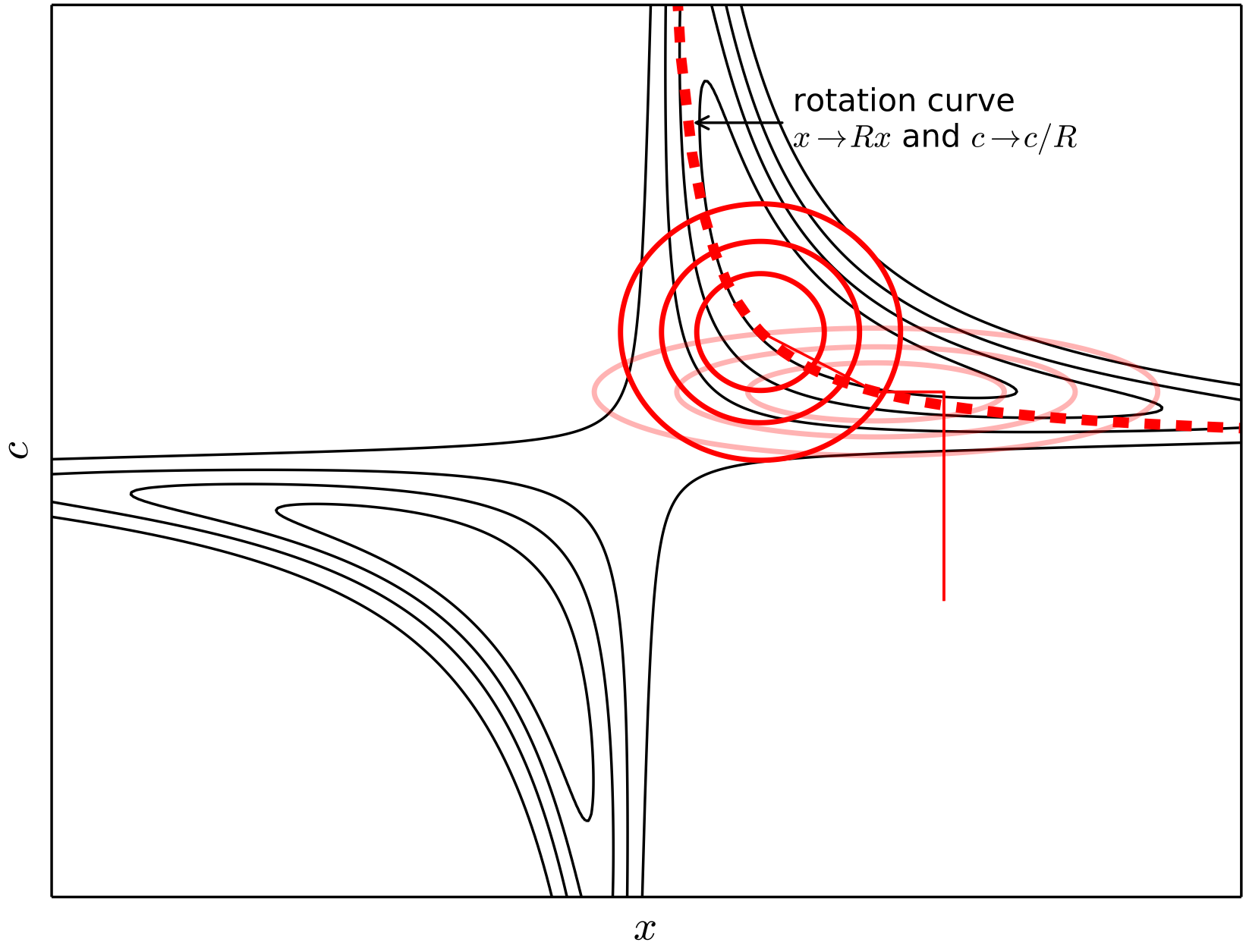
Iteration 1 - Update $q(x)$



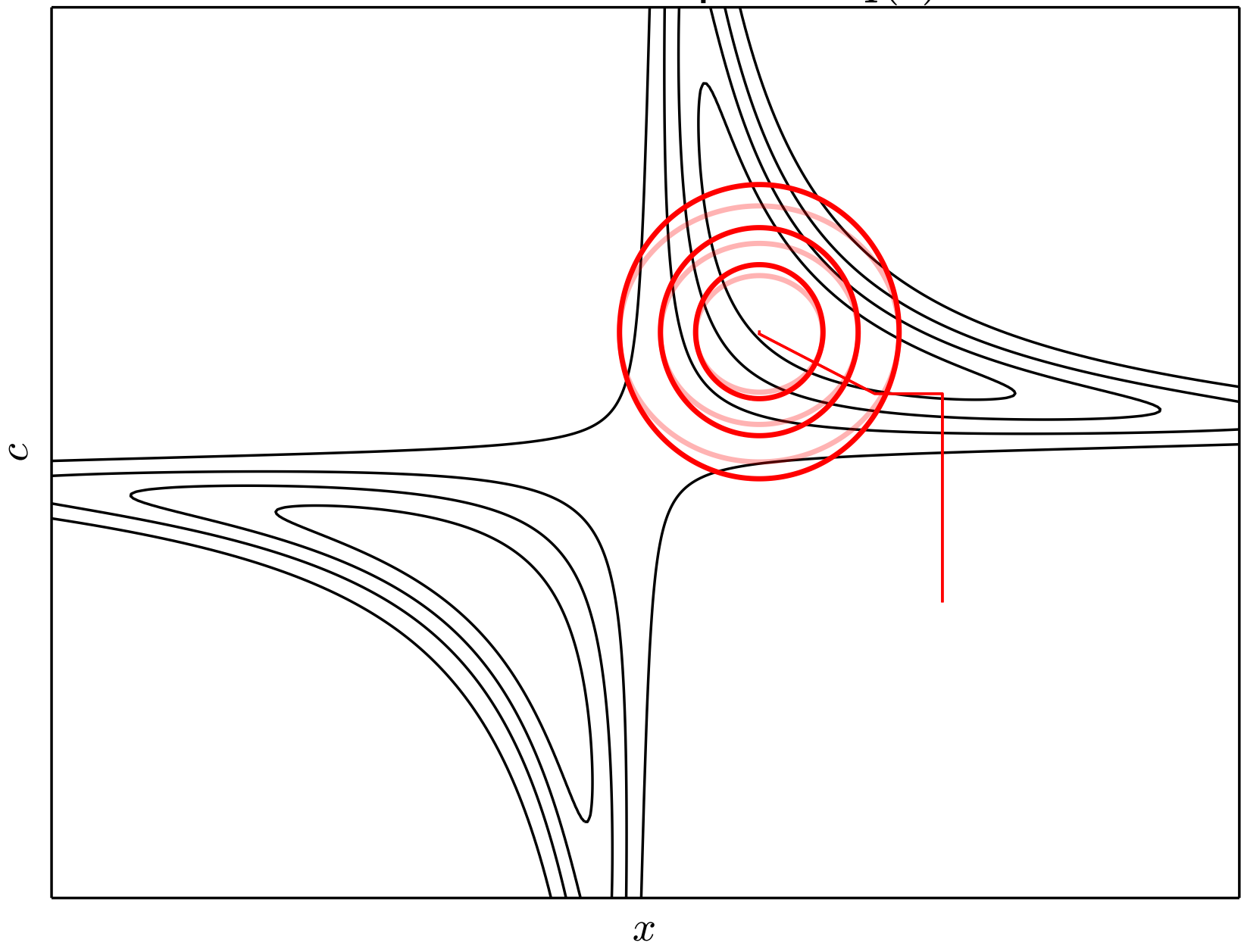
Iteration 1 - Rotation



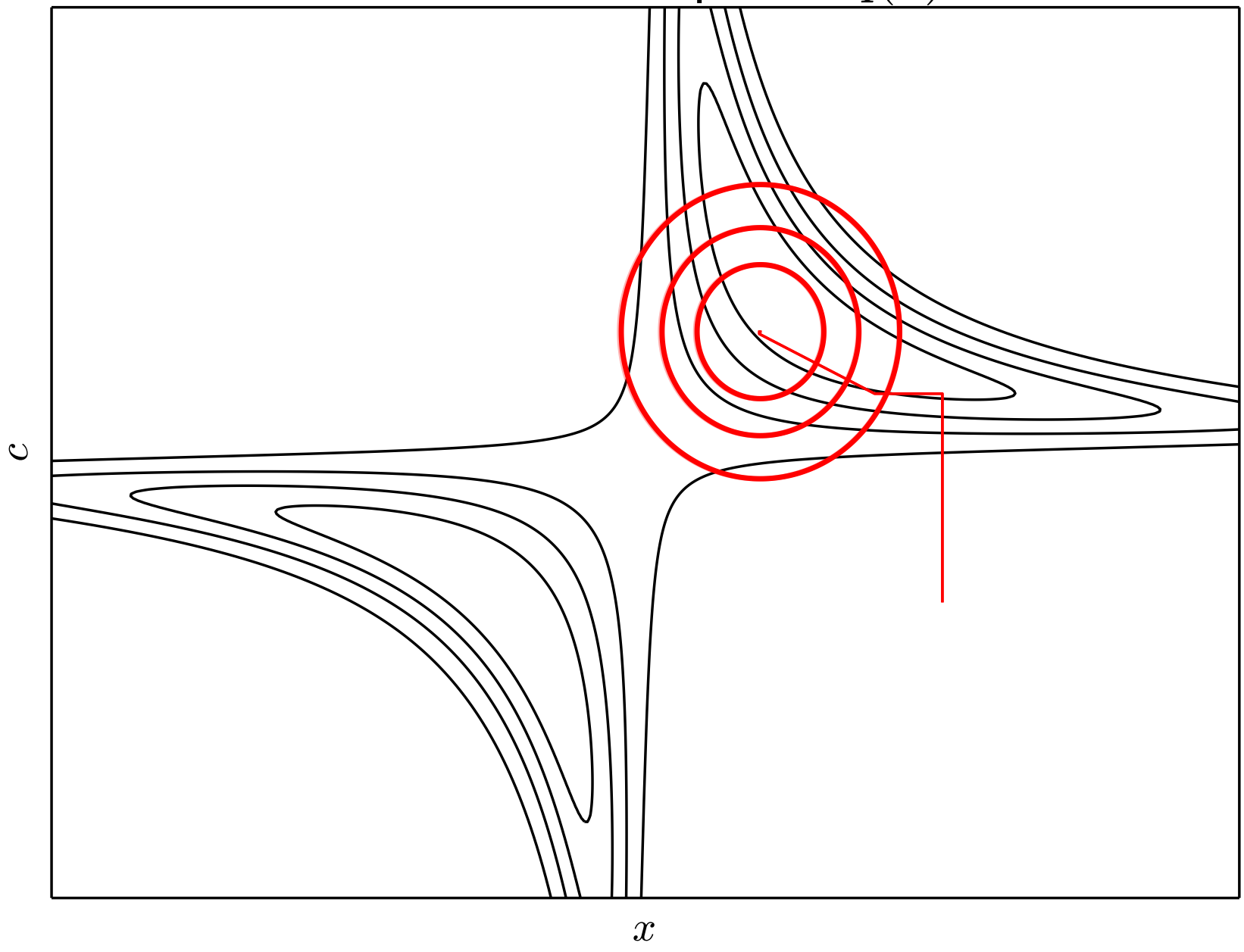
Iteration 1 - Rotation



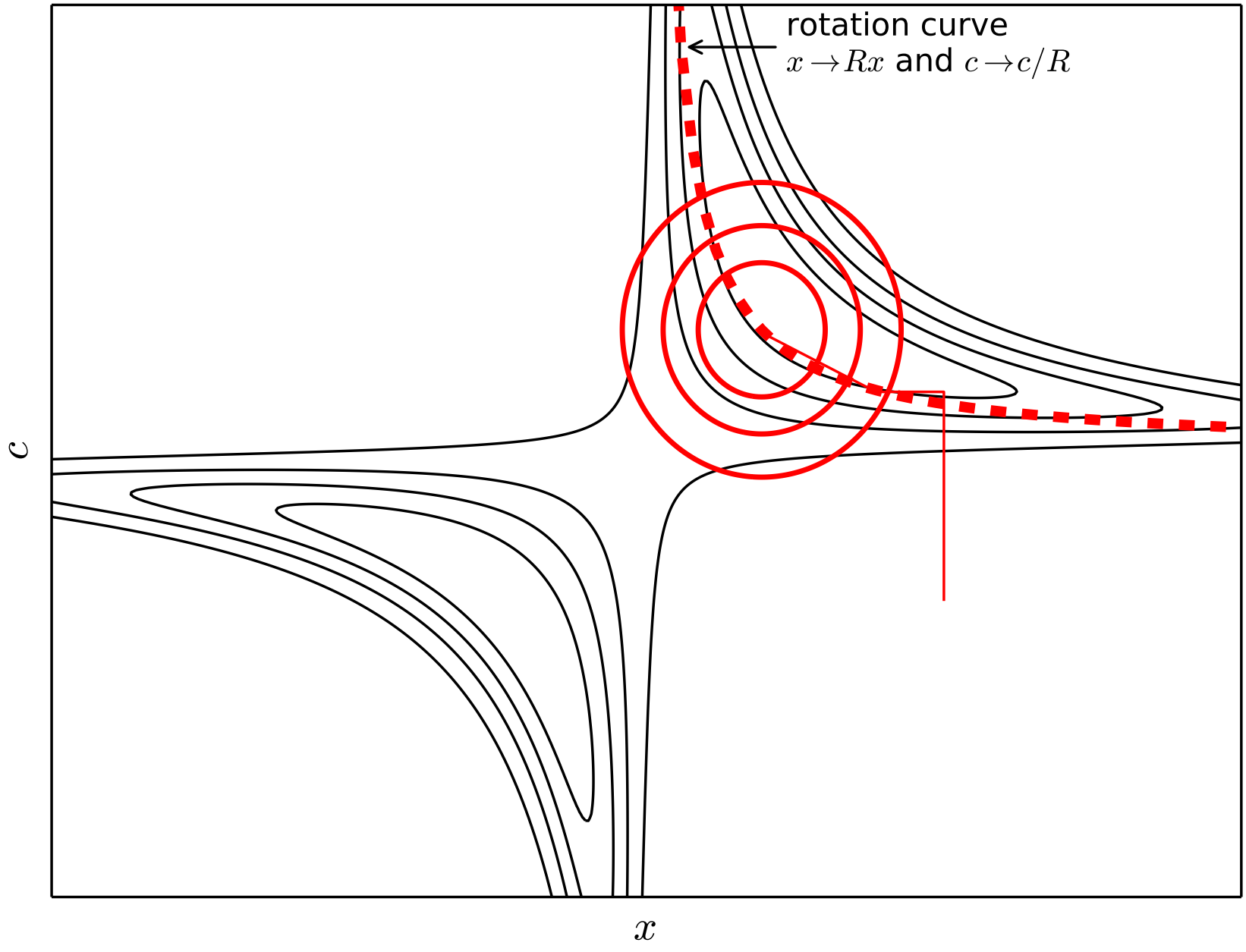
Iteration 2 - Update $q(c)$



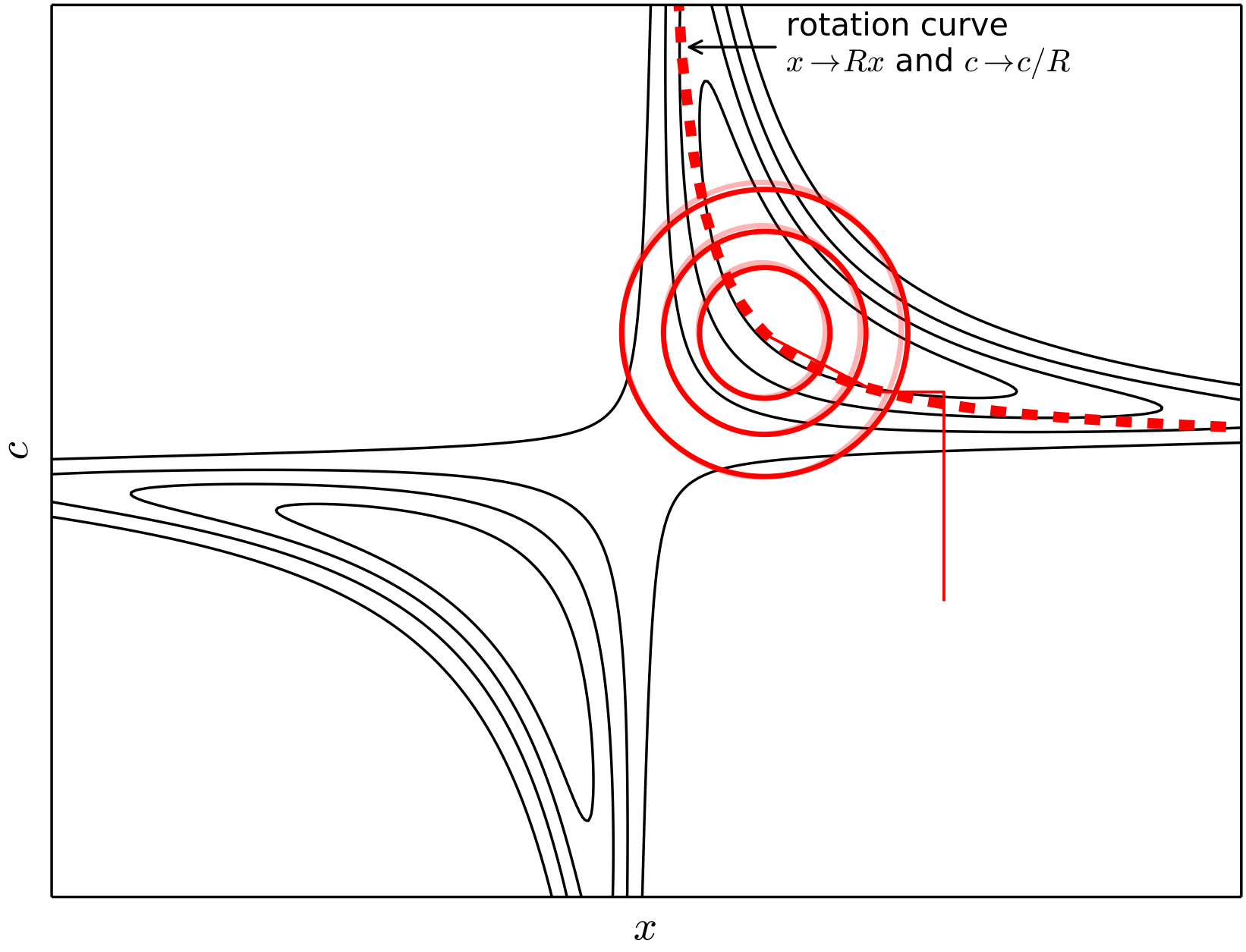
Iteration 2 - Update $q(x)$



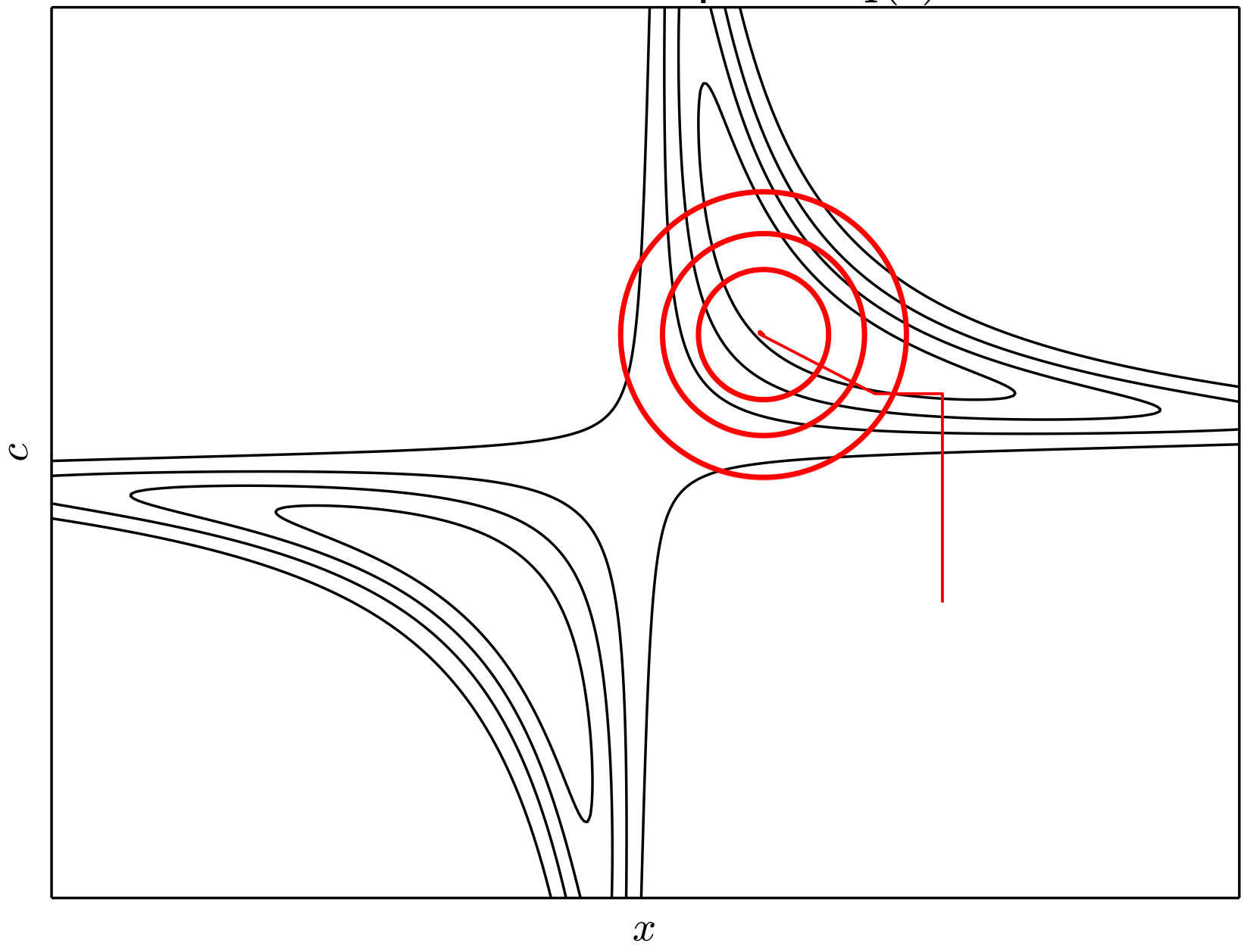
Iteration 2 - Rotation



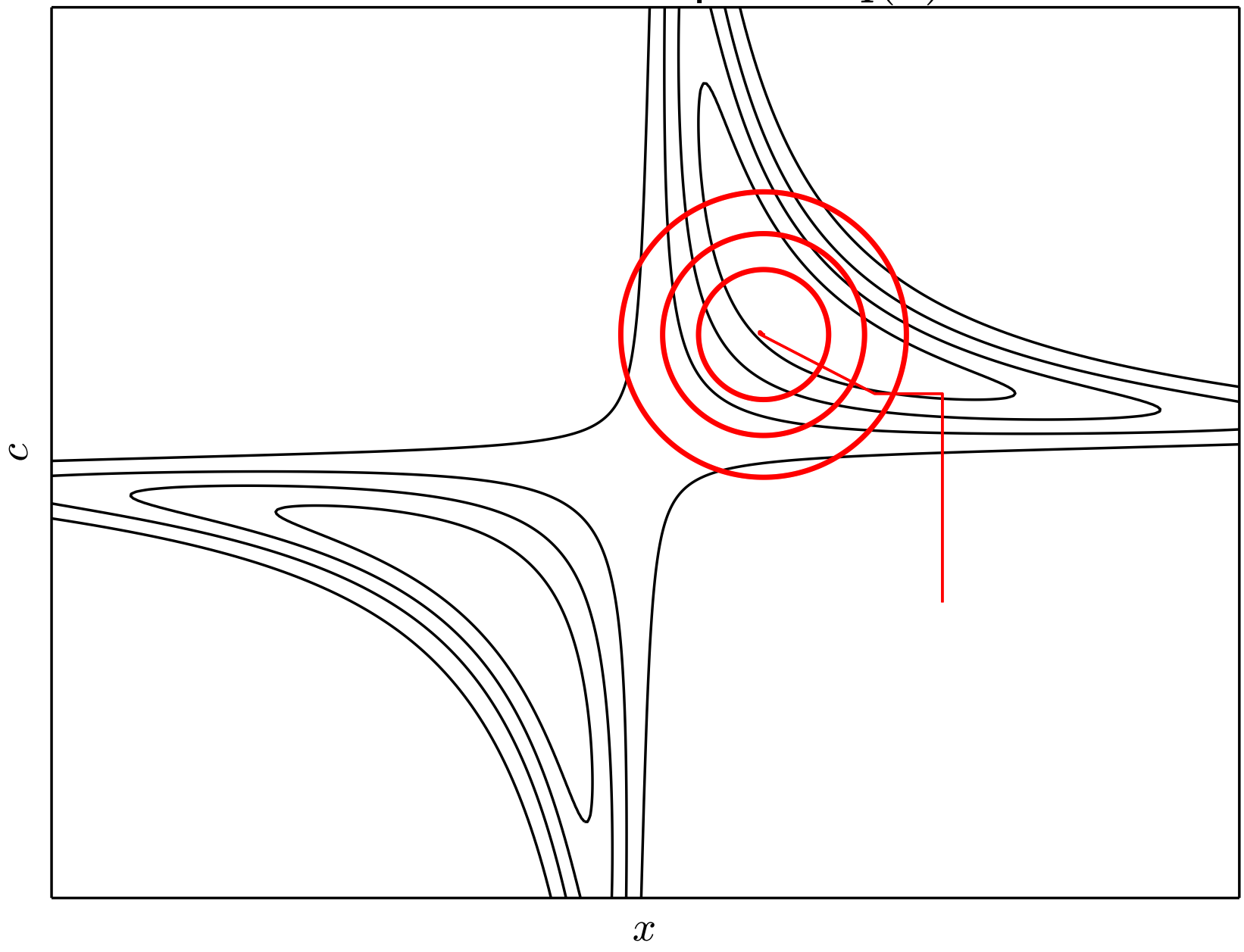
Iteration 2 - Rotation



Iteration 3 - Update $q(c)$

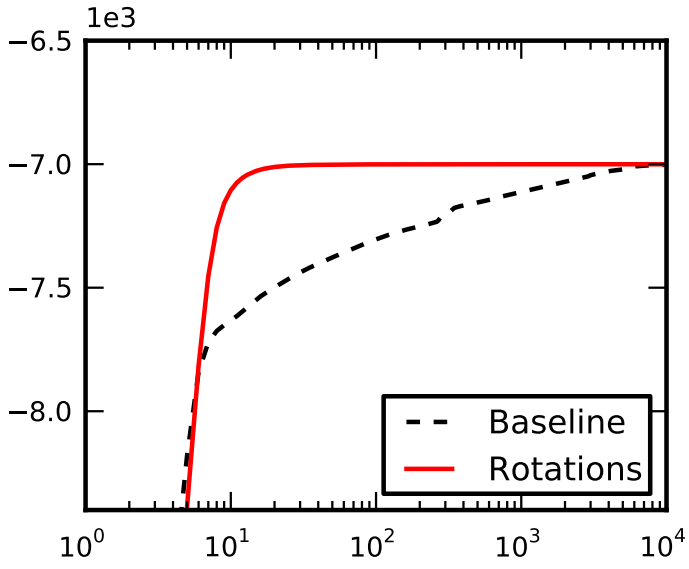


Iteration 3 - Update $q(x)$

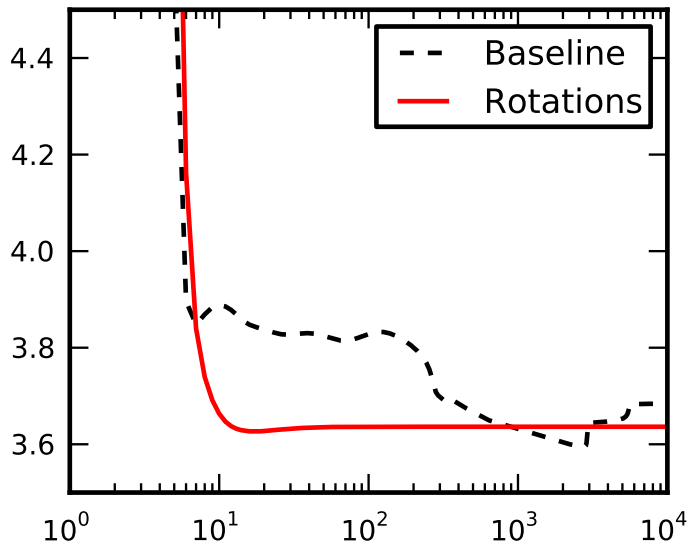


Artificial experiment

- 400 observations with 30 dimensions
- 8-dimensional latent space
- Performance as a function of VB iterations (log-scale):



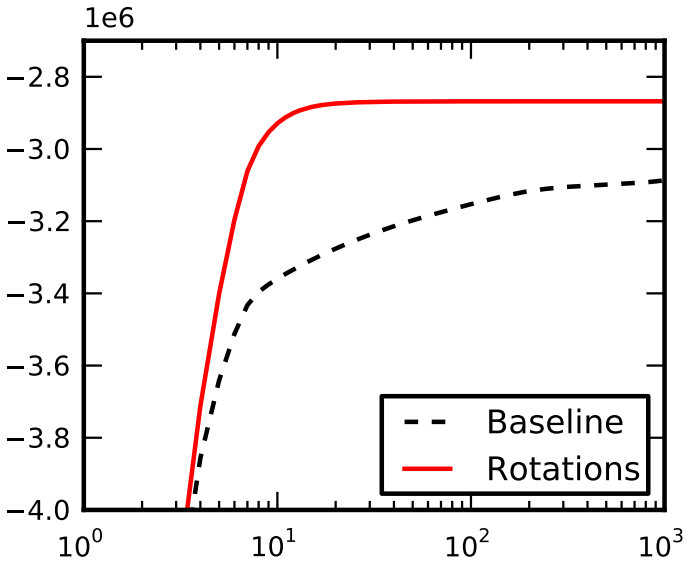
(a) VB lower bound



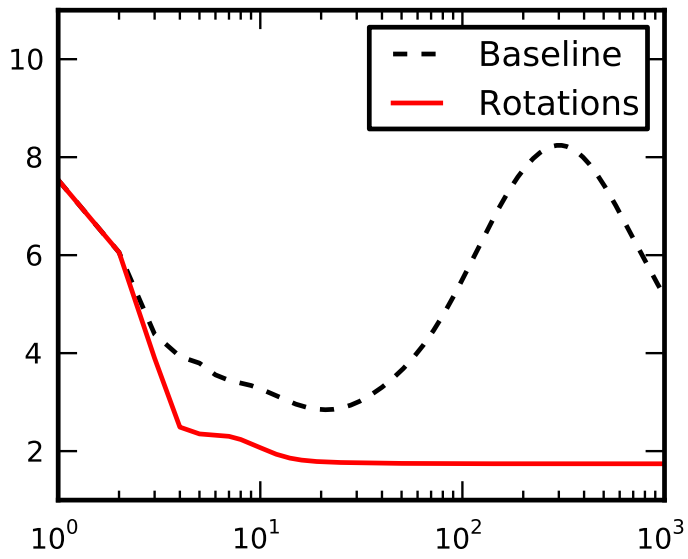
(b) Test RMSE

Weather data experiment

- 89202 observations with 66 dimensions
- 10-dimensional latent space
- Performance as a function of VB iterations (log-scale):



(a) VB lower bound



(b) Test RMSE

Summary

Rotation speeds up VB learning by orders of magnitude

Code and data available at
`http://users.ics.aalto.fi/jluttine/ecml2013`