Solving chaotic differential equations using an adaptative algorithm

Master Thesis

Lionel Walter

Prof. Alfio Quarteroni
Dr. Erik Burman
Dr. Paolo Zunino

Private Defense / January 27th 2005

Outline

Outline

Outline

The Cauchy Problem

Find a function $X : \mathbb{R} \to \mathbb{R}^d$ such that

$$\begin{cases} X'(t) = a(t, X(t)) & \forall t \in [0\ T] \\ X(0) = X_0 \end{cases}$$

$a(\cdot, \cdot)$ and X_0 are given.

Finding the exact X is usually impossible. We approximate X by $\bar{X}$ and our goal is to minimize

$$g(X(T)) - g(\bar{X}(T))$$

where $g : \mathbb{R}^d \to \mathbb{R}$ is a given function, called the goal function.

The Cauchy Problem

Find a function $X : \mathbb{R} \to \mathbb{R}^d$ such that

$$\begin{cases} X'(t) = a(t, X(t)) & \forall t \in [0\ T] \\ X(0) = X_0 \end{cases}$$

$a(\cdot, \cdot)$ and X_0 are given.

Finding the exact X is usually impossible. We approximate X by $\bar{X}$ and our goal is to minimize

$$g(X(T)) - g(\bar{X}(T))$$

where $g : \mathbb{R}^d \to \mathbb{R}$ is a given function, called the goal function.

The Cauchy Problem

Find a function $X : \mathbb{R} \to \mathbb{R}^d$ such that

$$\begin{cases} X'(t) = a(t, X(t)) & \forall t \in [0\ T] \\ X(0) = X_0 \end{cases}$$

$a(\cdot, \cdot)$ and X_0 are given.

Finding the exact X is usually impossible. We approximate X by $\bar{X}$ and our goal is to minimize

$$g(X(T)) - g(\bar{X}(T))$$

where $g : \mathbb{R}^d \to \mathbb{R}$ is a given function, called the goal function.

Goal over the entire time interval

Assume we want to have a goal function of the following form :

$$h(X(T)) + \int_0^T k(t, X(t))dt$$

for given functions $h : \mathbb{R}^d \to \mathbb{R}$ et $k : \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}$.

If we modify the Cauchy problem as follows

$$\begin{cases} X'(t) = a(t, X(t)) \quad \forall t \in [0\ T] \\ X(0) = X_0 \\ y'(t) = k(t, y(t)) \\ y(0) = 0 \end{cases}$$

then we can define the goal $g(X(T)) = h(X(T)) + y(T)$ and we get the wanted goal.

Goal over the entire time interval

Assume we want to have a goal function of the following form :

$$h(X(T)) + \int_0^T k(t, X(t))dt$$

for given functions $h : \mathbb{R}^d \to \mathbb{R}$ et $k : \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}$.
If we modify the Cauchy problem as follows

$$\begin{cases} X'(t) = a(t, X(t)) \quad \forall t \in [0\ T] \\ X(0) = X_0 \\ y'(t) = k(t, y(t)) \\ y(0) = 0 \end{cases}$$

then we can define the goal $g(X(T)) = h(X(T)) + y(T)$ and we get the wanted goal.

What is Chaos ?

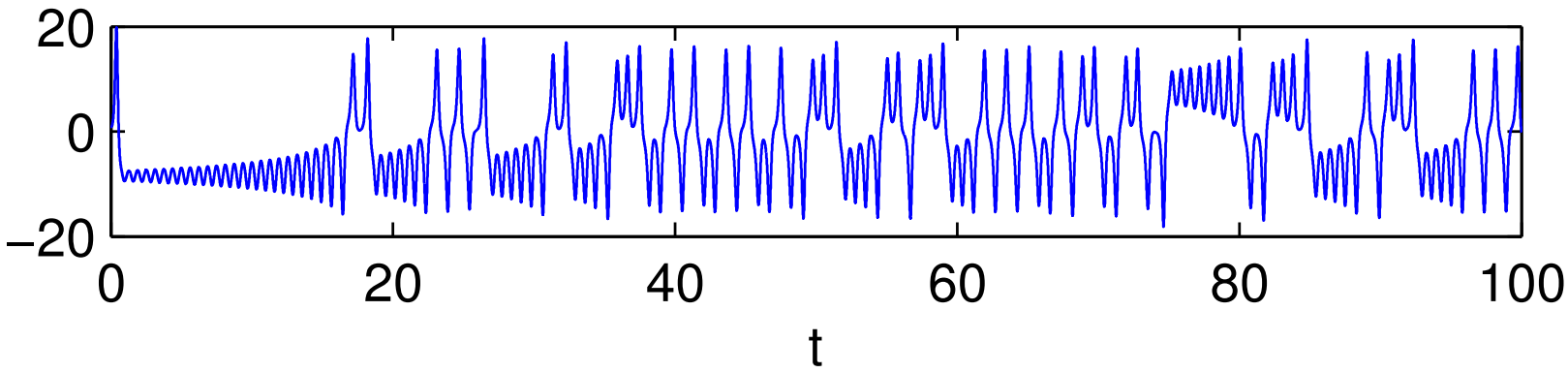
Chaos is used to indicate behavior of solutions to dynamical systems that is highly irregular and usually unexpected.

$\rightsquigarrow$ Lorenz applet.

What is Chaos ?

Chaos is used to indicate behavior of solutions to dynamical systems that is highly irregular and usually unexpected.

$\rightsquigarrow$ Lorenz applet.



This intuitive notion is present in religion, philosophy, politics, physics. . .

And in Mathematics ?

It's rather an idea than a precise mathematical concept. But Edward Ott gives a mathematical definition of Chaos. He says that if the difference between the solution of an ODE and the solution of the same ODE slightly modified grows exponentially with time then the system is said to be chaotic.

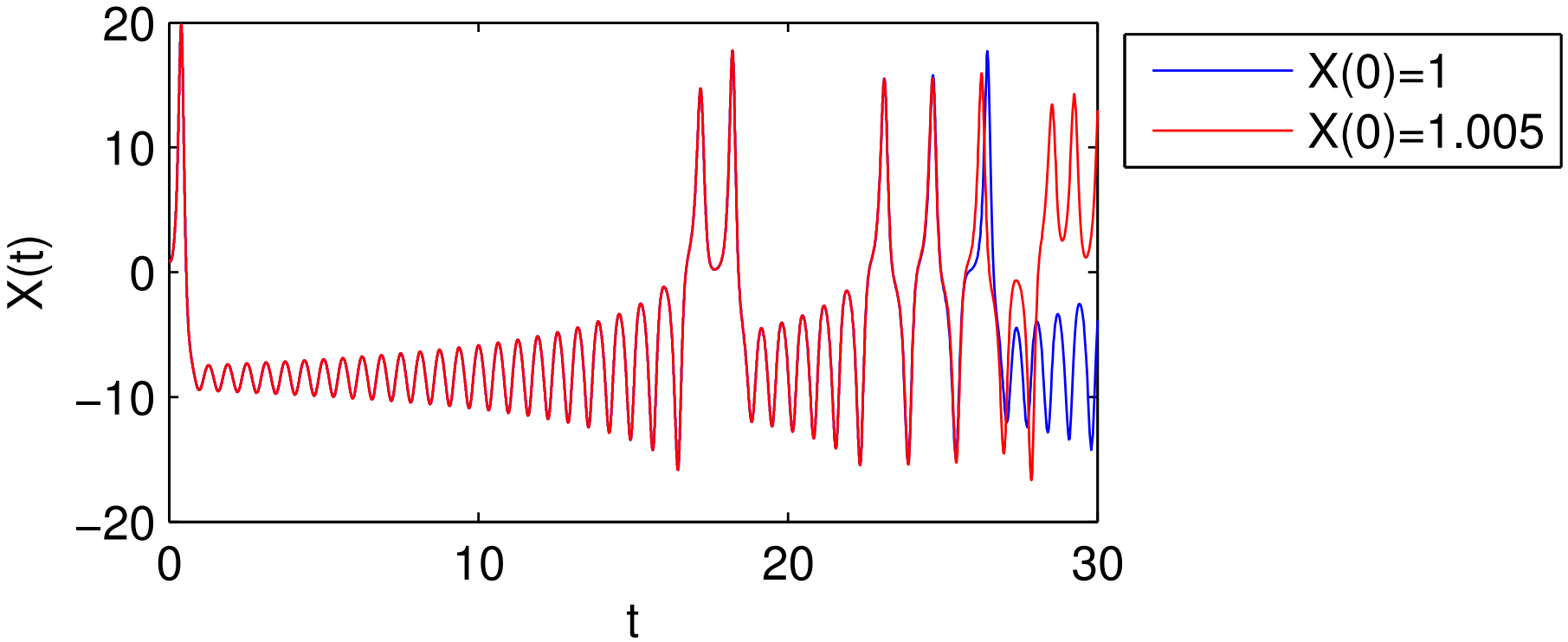
$$\begin{cases} X'(t) = a(t, X(t)) & \forall t \in [0\ T] \\ X(0) = X_0 \end{cases} \qquad \begin{cases} Y'(t) = a(t, Y(t)) & \forall t \in [0\ T] \\ Y(0) = X_0 + \delta_0 \end{cases}$$

If

$$\lim_{\delta_0 \to 0} \frac{\|Y(t) - X(t)\|}{\delta_0} \approx \mathrm{e}^{Dt}$$

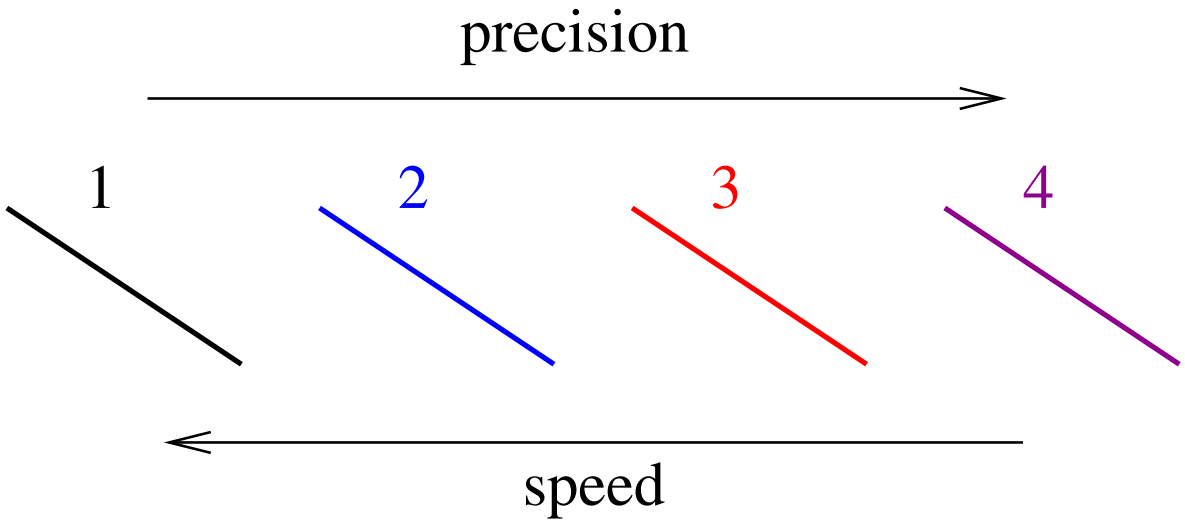
for a constant $D > 0$, then the system is said to be chaotic.

Example



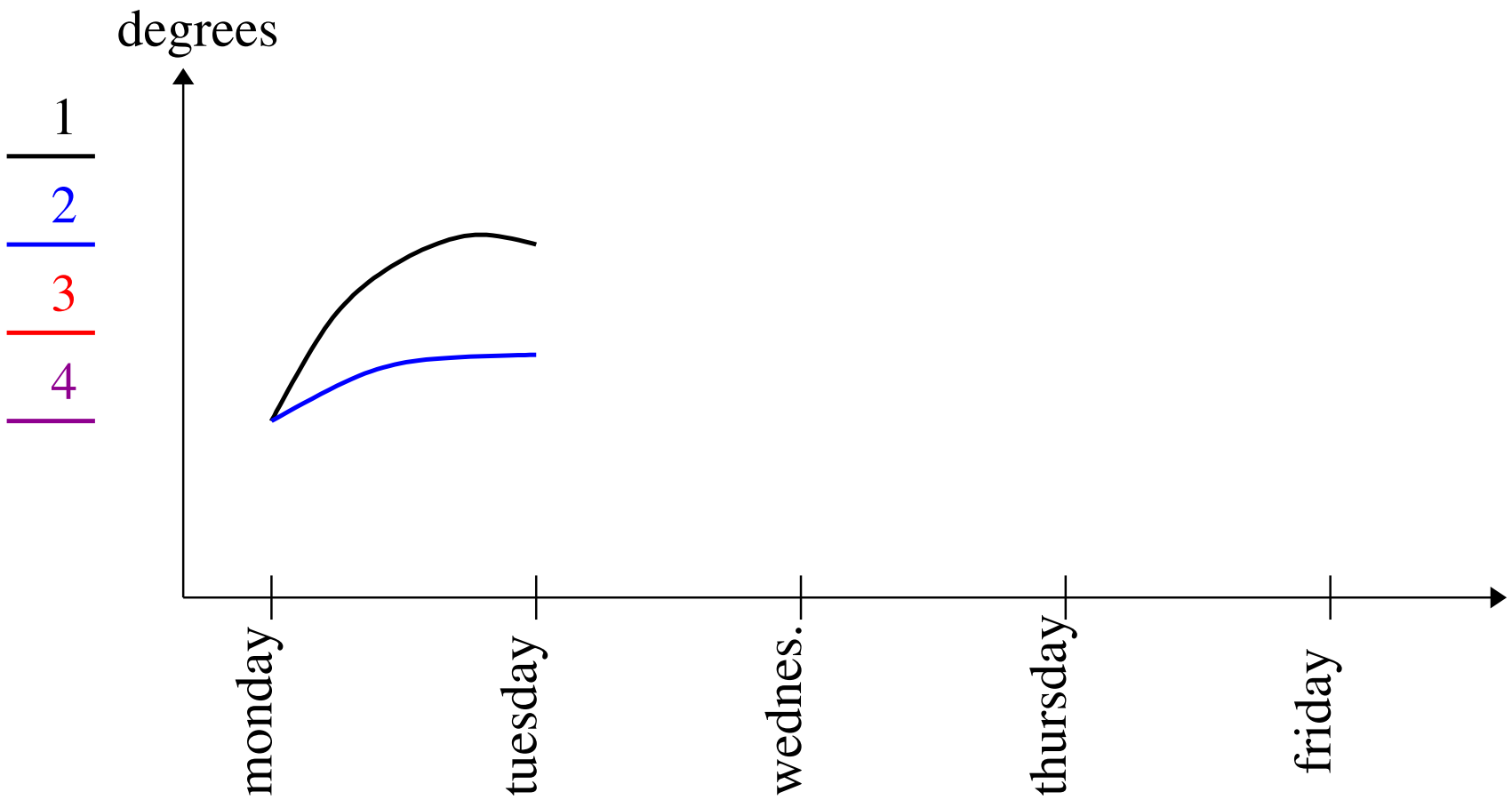
Meteorological Example

- Monday Morning, 10 degrees. What will be the temperature on Friday Morning ?
- We have all the laws of meteorology at hand.
- We have 4 methods at our disposal to compute the temperature of the following day.

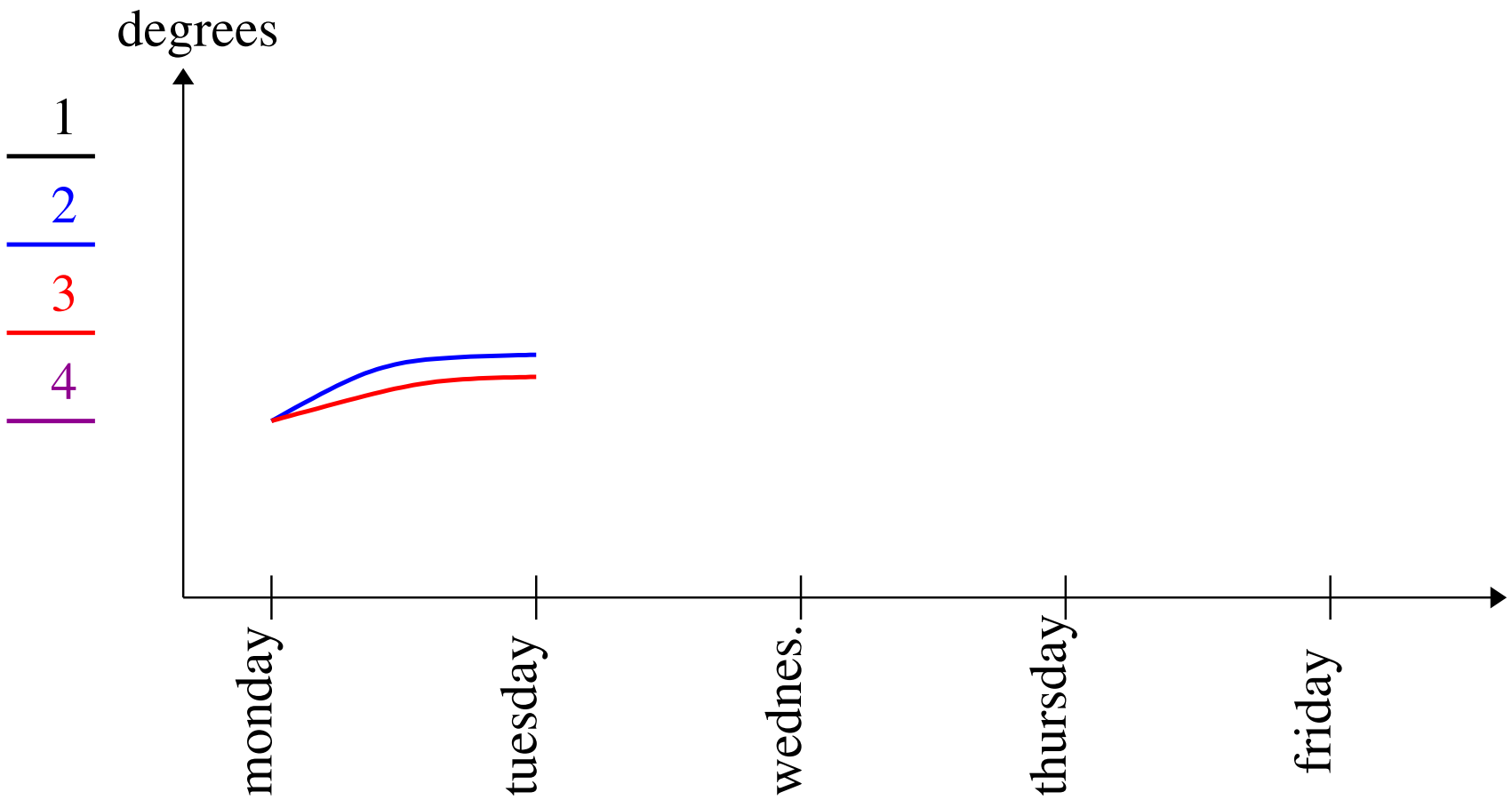


- We want to have Friday's temperature with a precision of 1 degree in the least amount of time

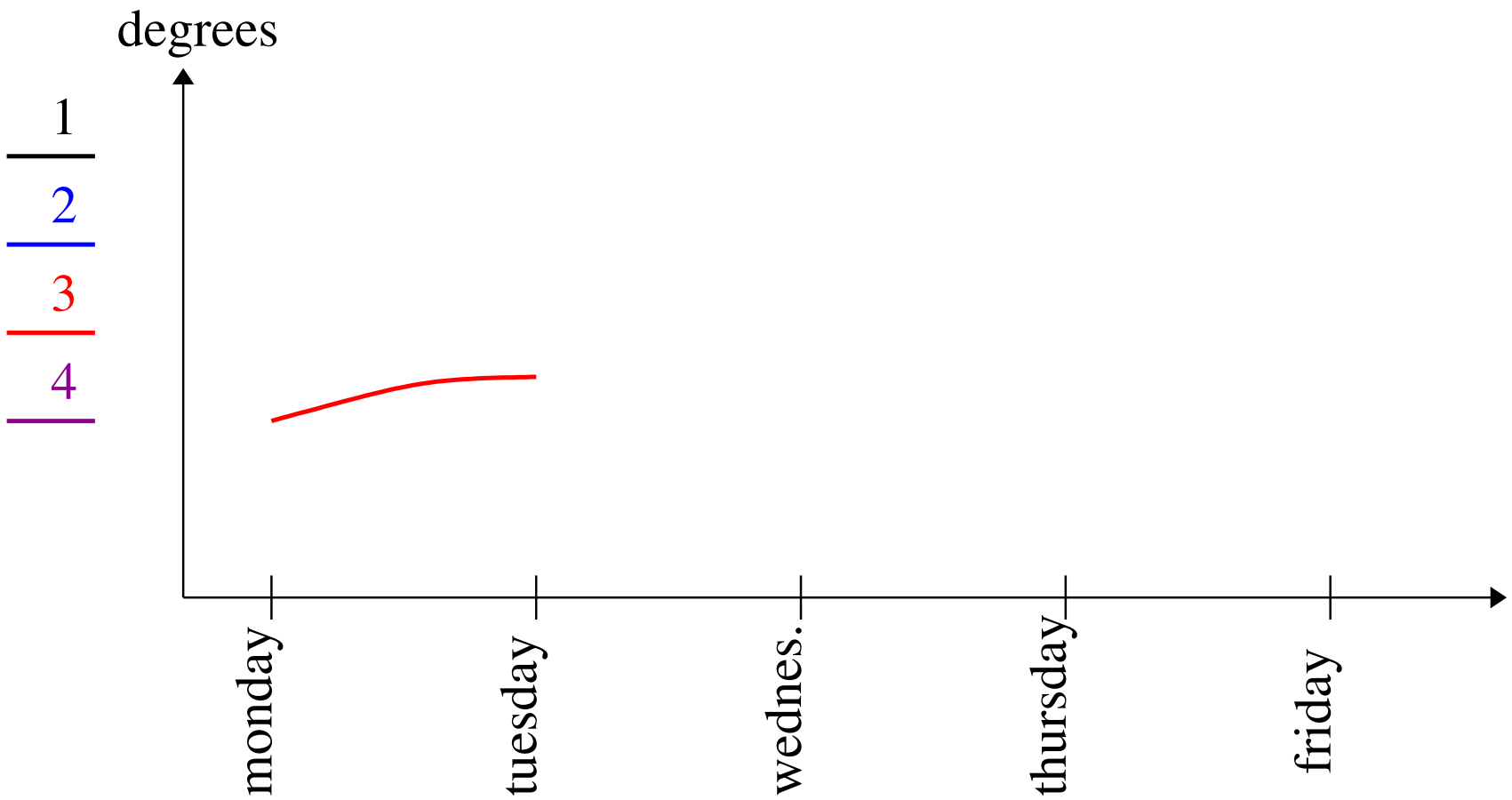
Strategy A



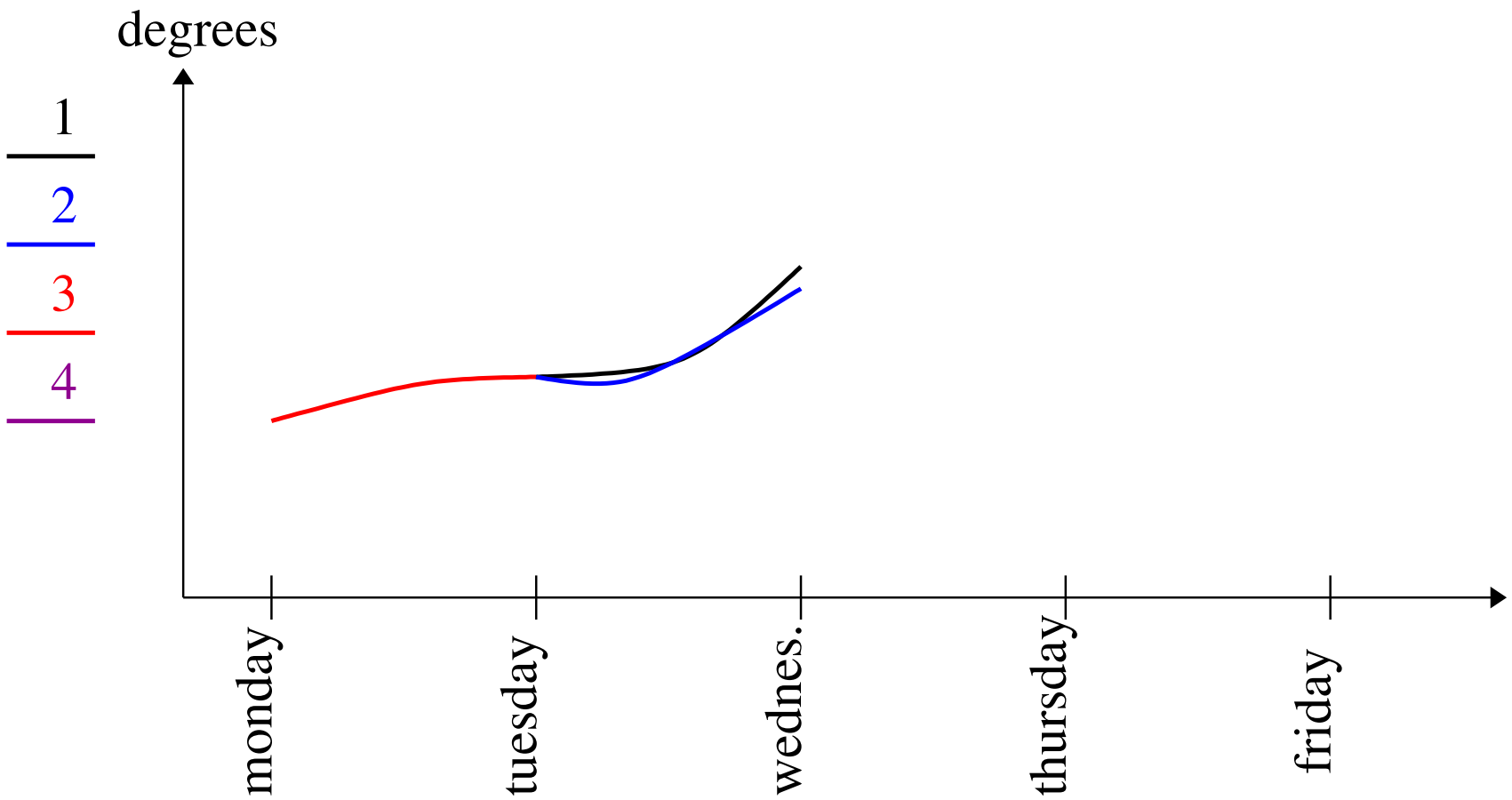
Strategy A



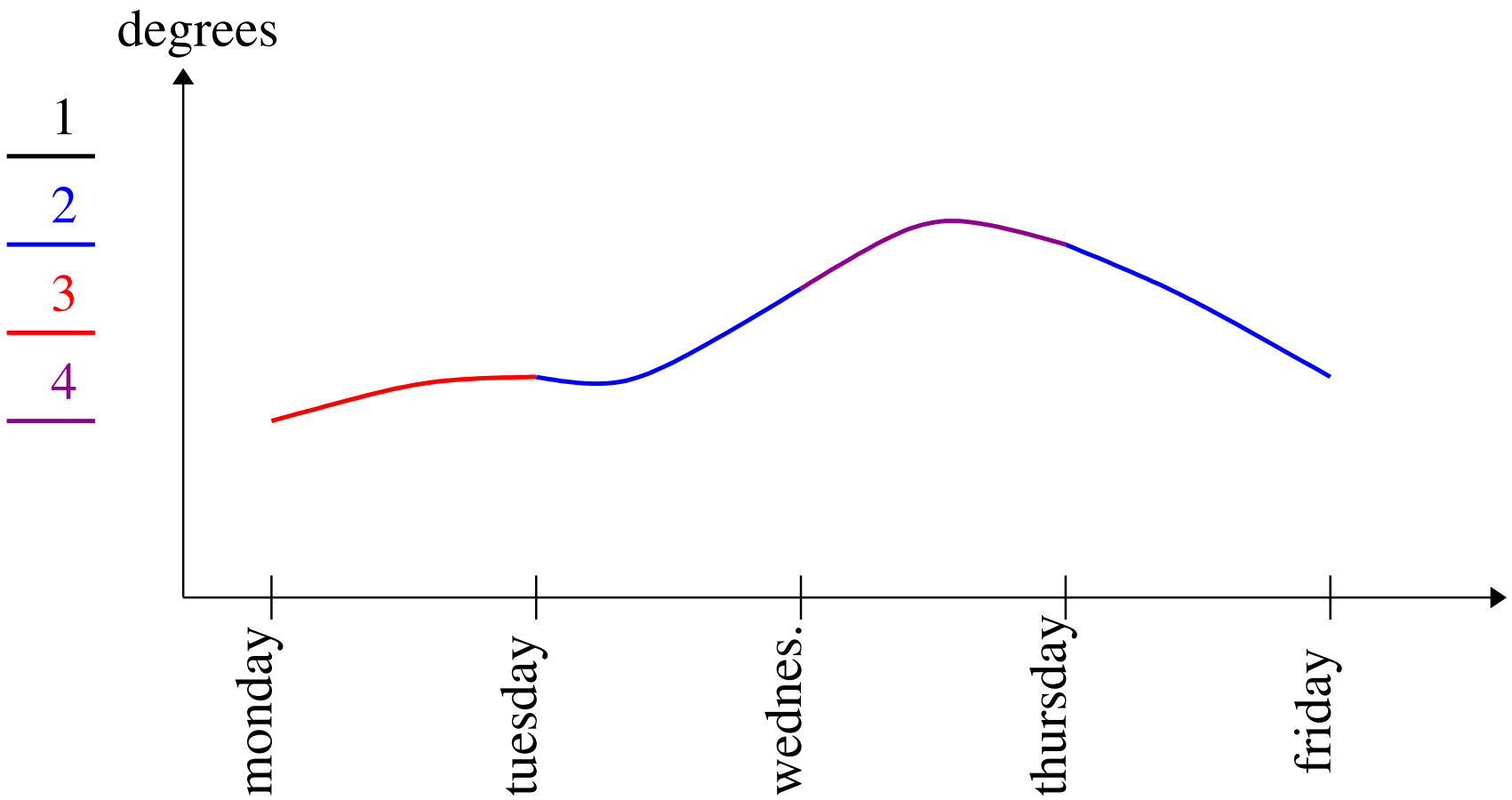
Strategy A



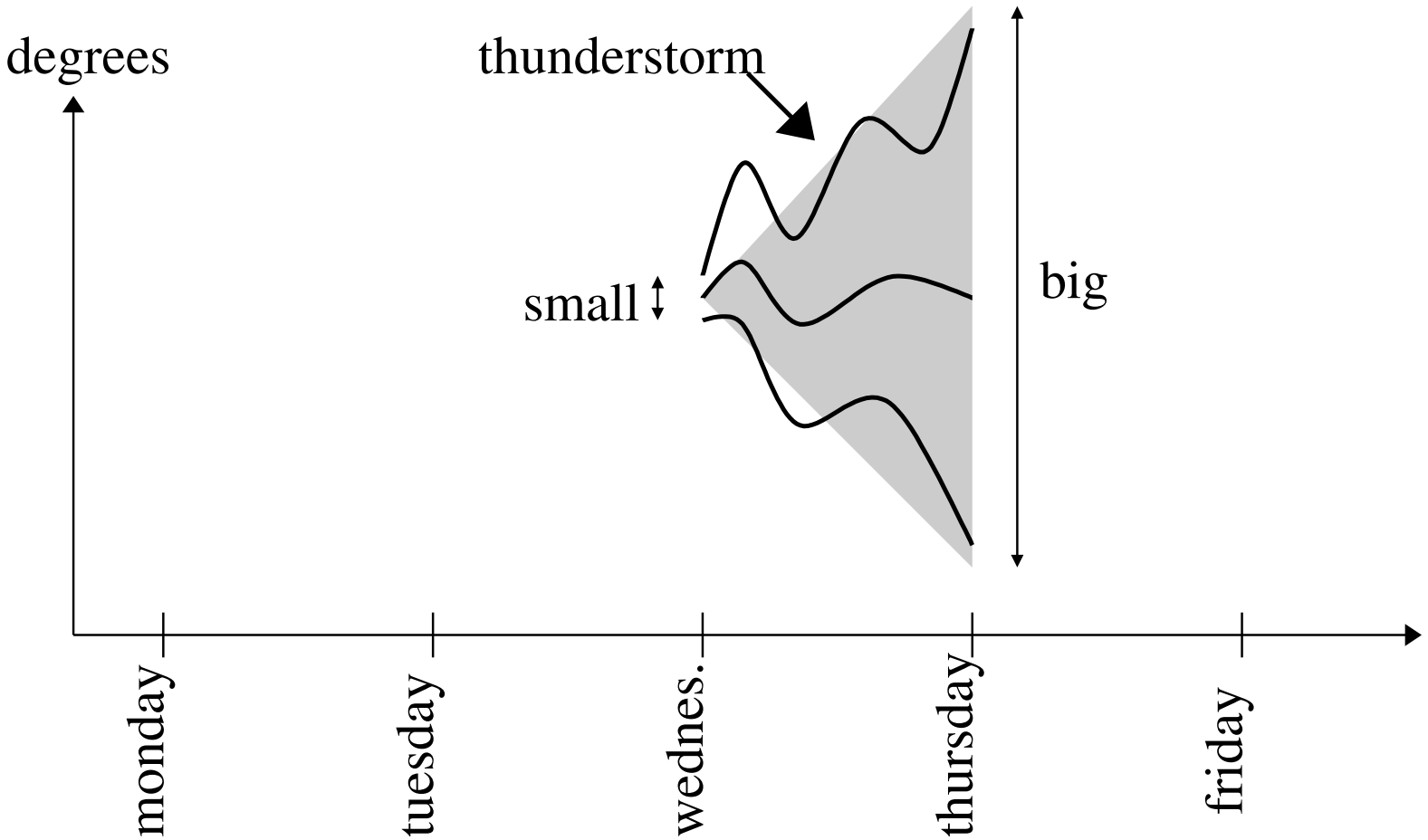
Strategy A



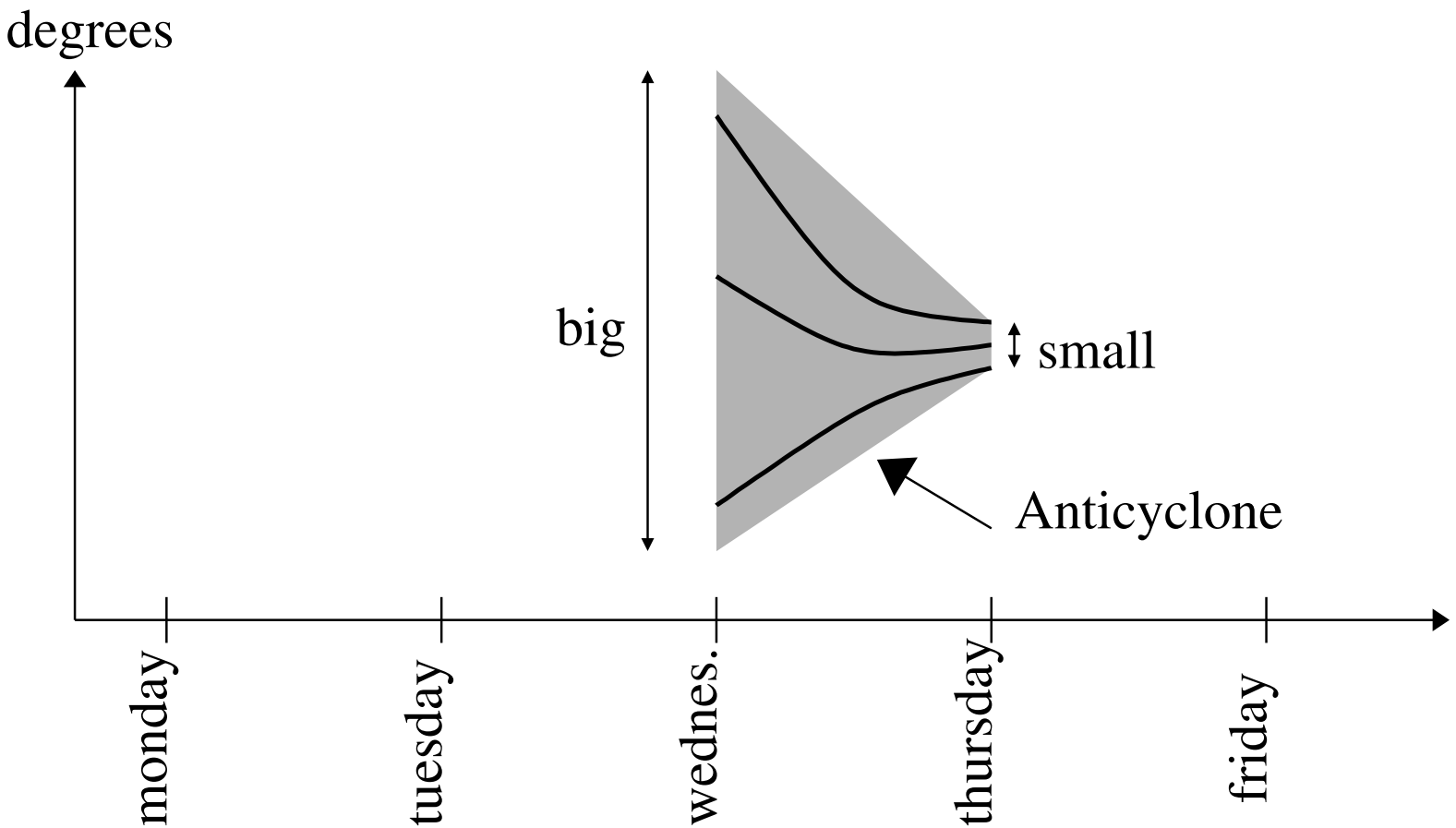
Strategy A



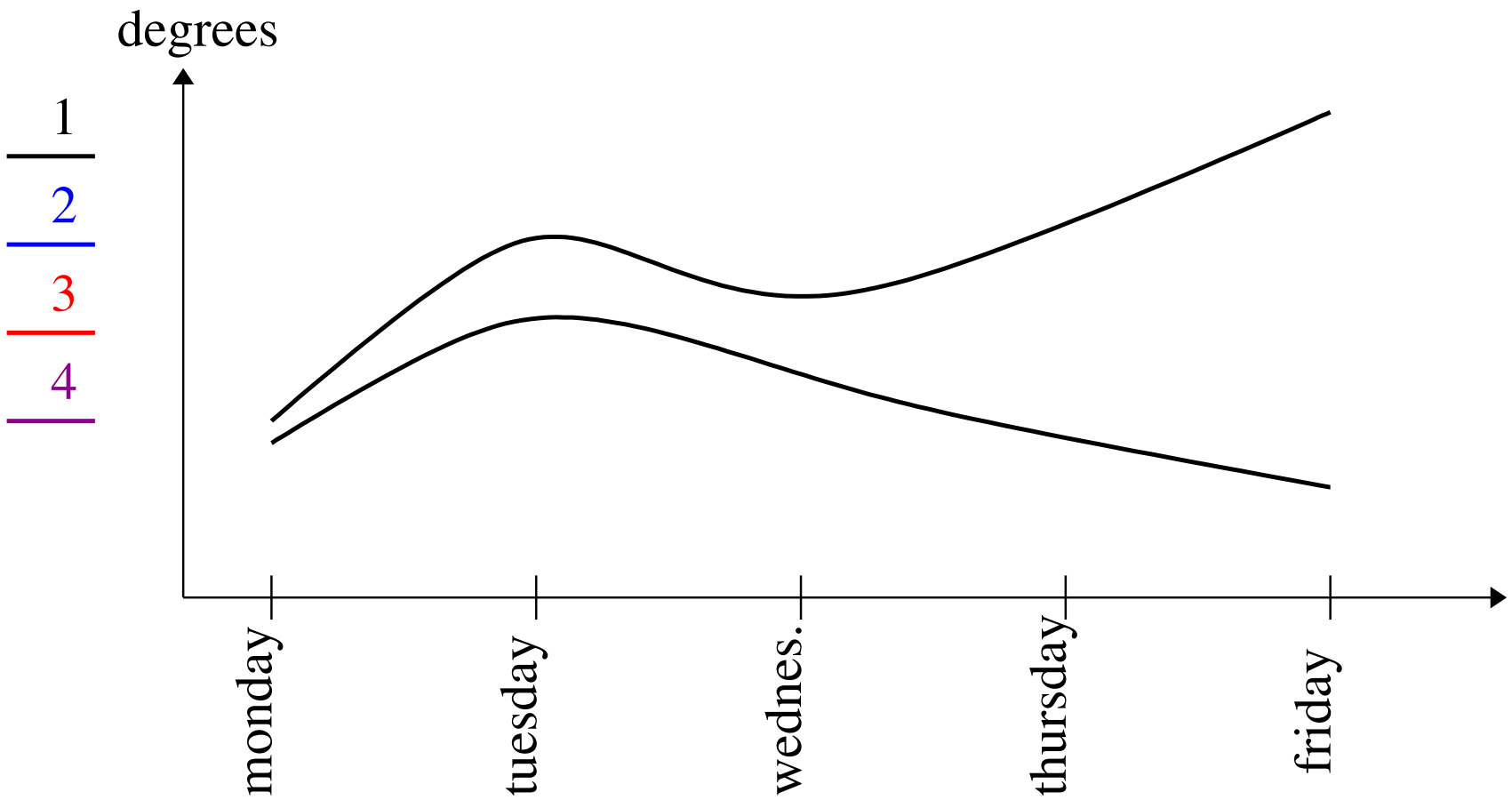
Chaotic problems : Thunderstorm



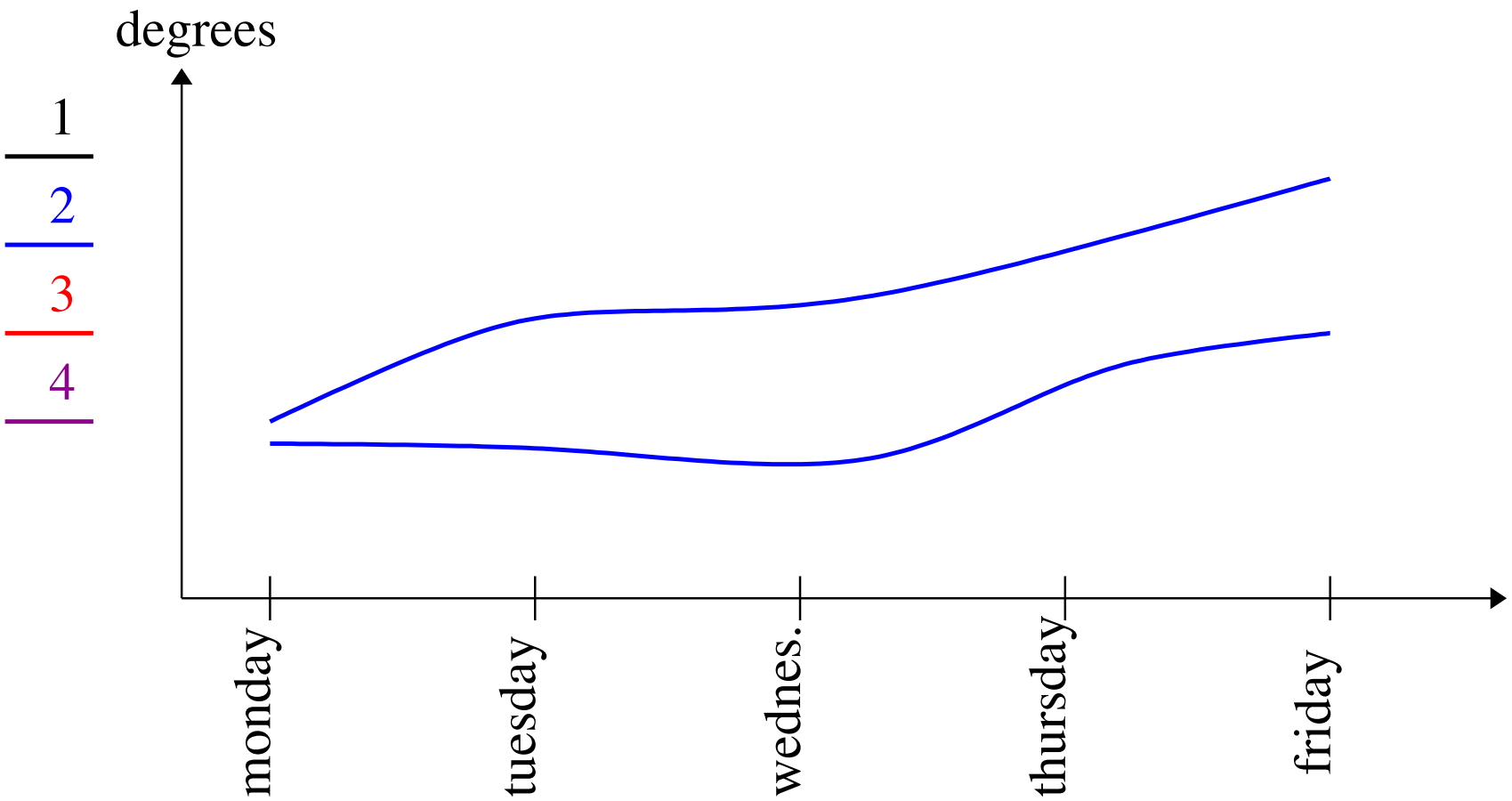
Chaotic problems : Anticyclone



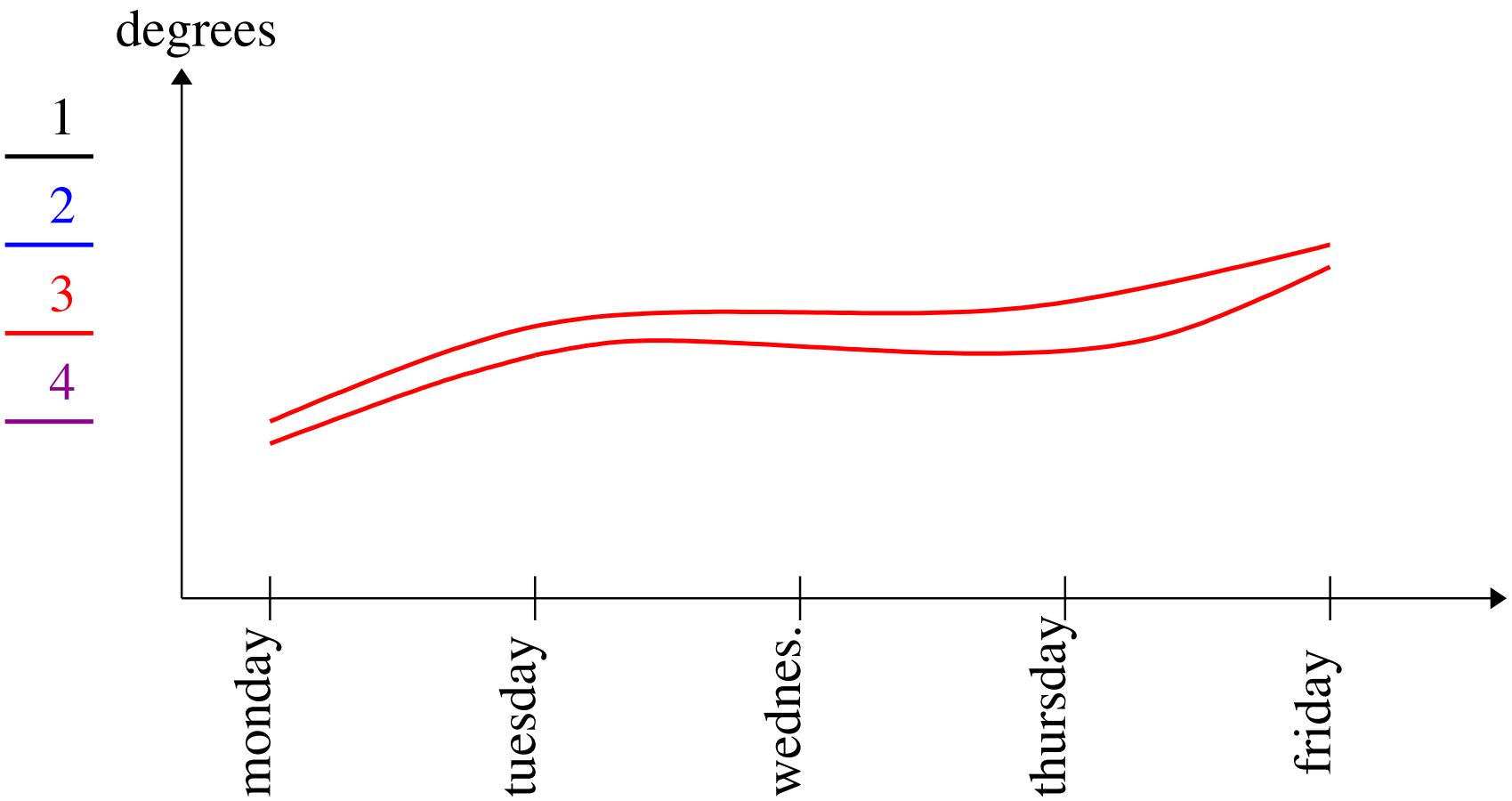
A better strategy : Strategy B



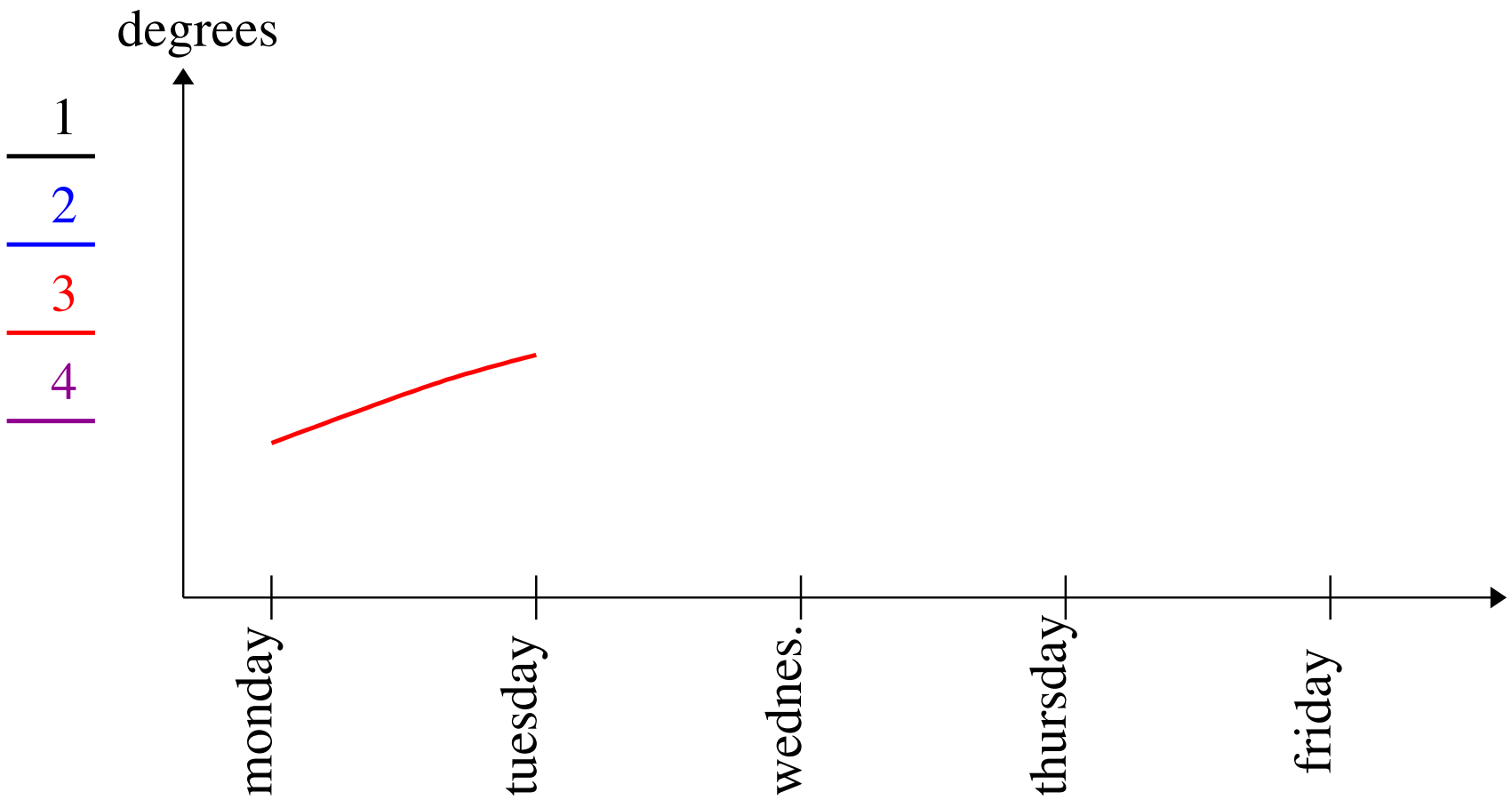
A better strategy : Strategy B



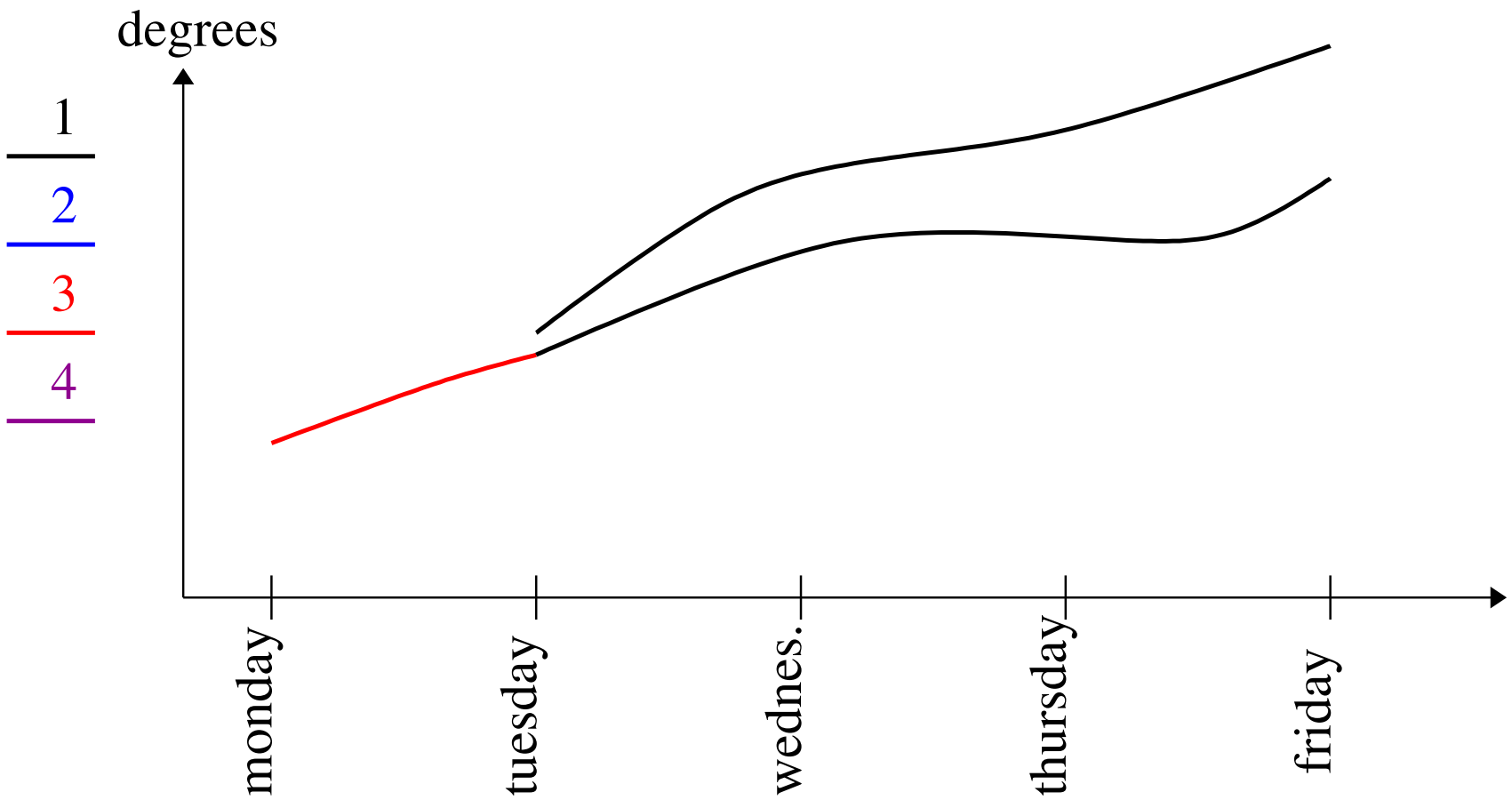
A better strategy : Strategy B



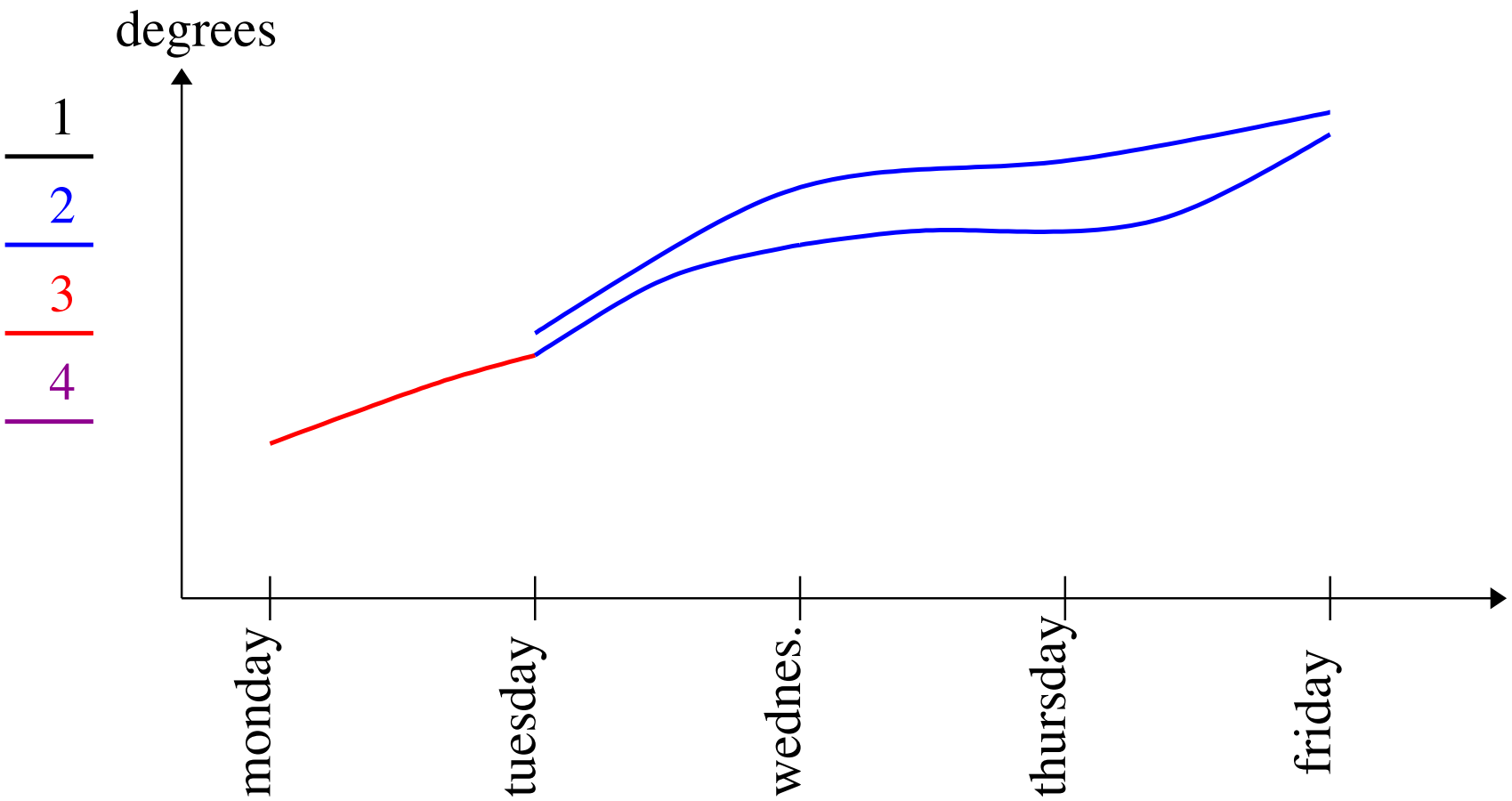
A better strategy : Strategy B



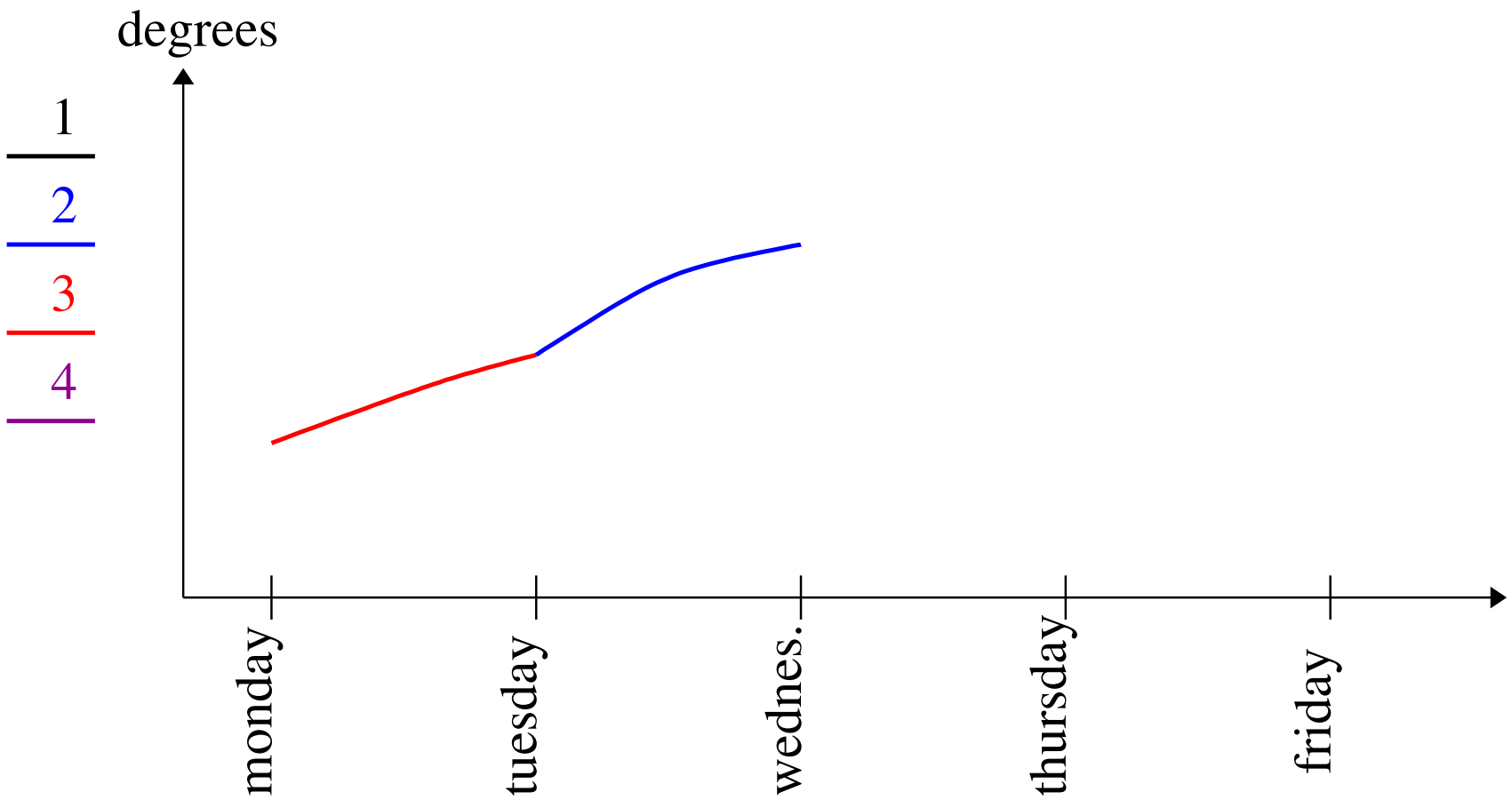
A better strategy : Strategy B



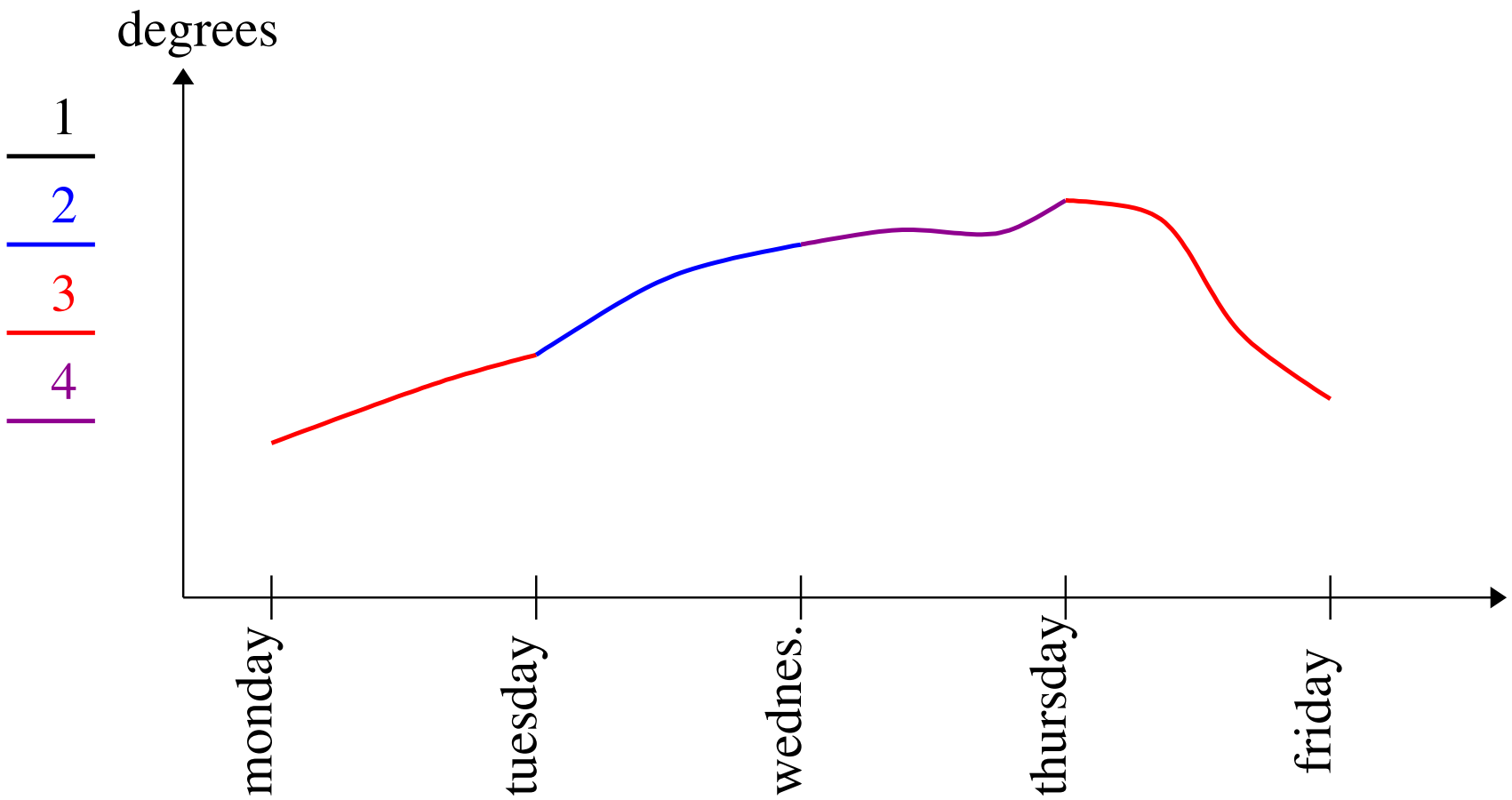
A better strategy : Strategy B



A better strategy : Strategy B



A better strategy : Strategy B



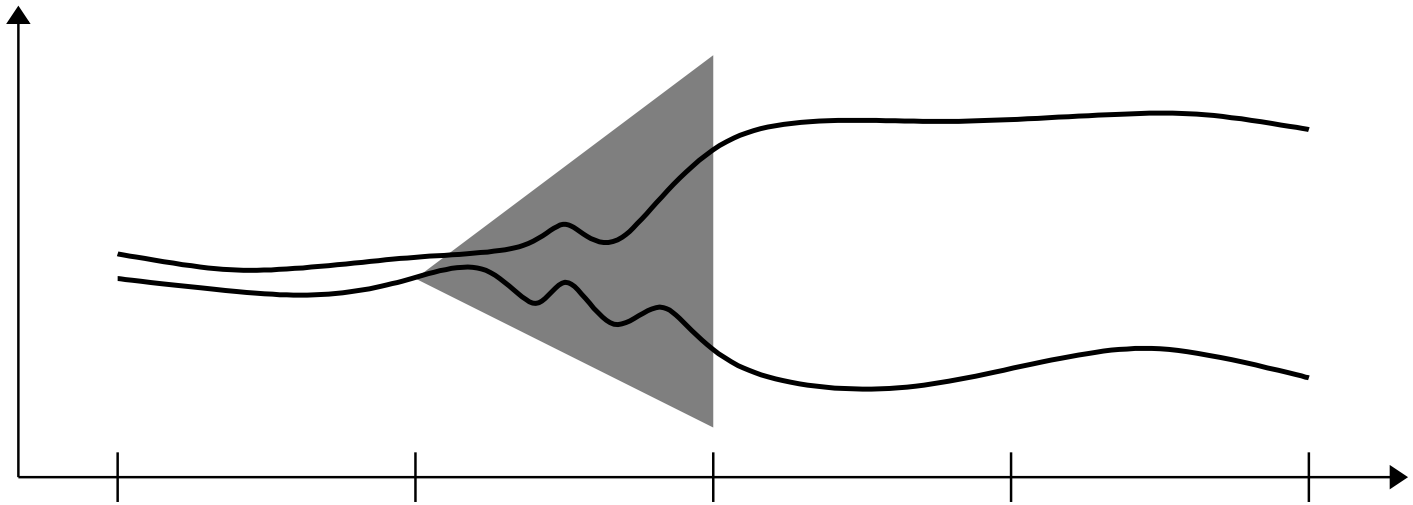
Analysis of Strategy B

Advantage

Strategy B deals with Thunderstorms and Anticyclones.

Drawbacks

- Strategy B is much slower than strategy A
- Strategy B could give wrong results



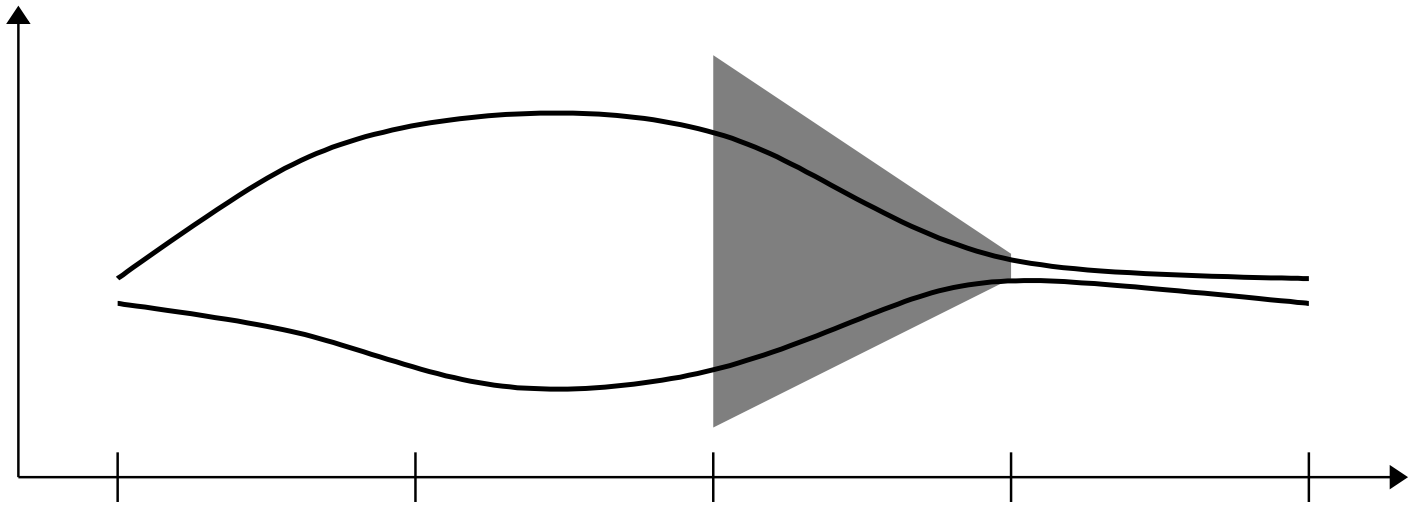
Analysis of Strategy B

Advantage

Strategy B deals with Thunderstorms and Anticyclones.

Drawbacks

- Strategy B is much slower than strategy A
- Strategy B could give wrong results



Analysis of Strategy B

Advantage

Strategy B deals with Thunderstorms and Anticyclones.

Drawbacks

- Strategy B is much slower than strategy A
- Strategy B could give wrong results

Analysis of Strategy B

Advantage

Strategy B deals with Thunderstorms and Anticyclones.

Drawbacks

- Strategy B is much slower than strategy A
- Strategy B could give wrong results

Solution

The central theorem says that the strategy B can be computed with the same amount of time as Strategy A

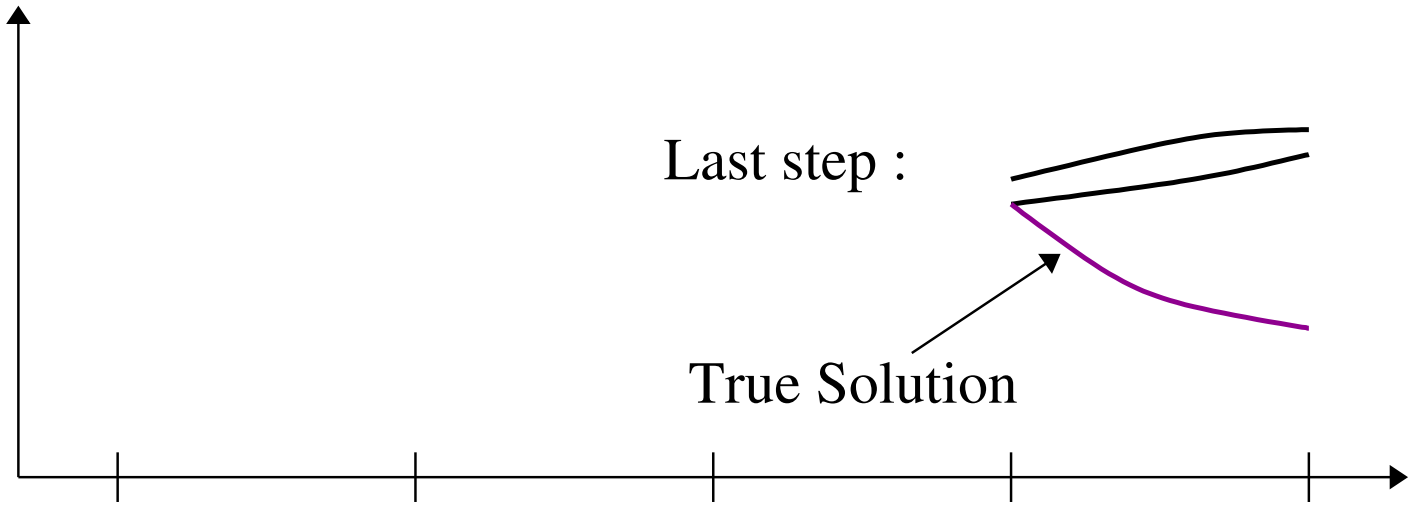
Analysis of Strategy B

Advantage

Strategy B deals with Thunderstorms and Anticyclones.

Drawbacks

- Strategy B is much slower than strategy A
- Strategy B could give wrong results



Analysis of Strategy B

Advantage

Strategy B deals with Thunderstorms and Anticyclones.

Drawbacks

- Strategy B is much slower than strategy A
- Strategy B could give wrong results

Solution

Combine Strategy A and Strategy B !

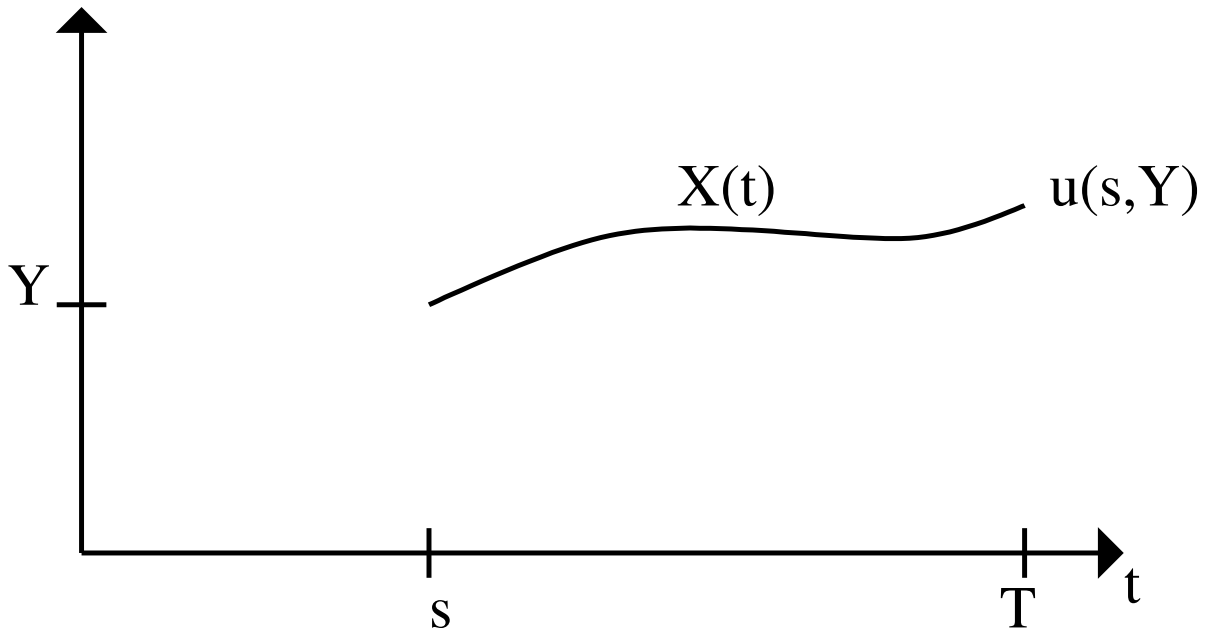
Conclusion

Combining strategy A and B and using the central theorem, we can now estimate temperature precisely, coping with thunderstorms and anticyclones. The time we need will be at most twice the time of strategy A only.

u and ψ

Definition (u)

$u(s, Y) \in \mathbb{R}$ is the value of $g(X(T))$ if we replace the initial condition $X(0) = X_0$ by $X(s) = Y$.



u if $g(x) = x$

u and ψ

Definition (u)

$u(s, Y) \in \mathbb{R}$ is the value of $g(X(T))$ if we replace the initial condition $X(0) = X_0$ by $X(s) = Y$.

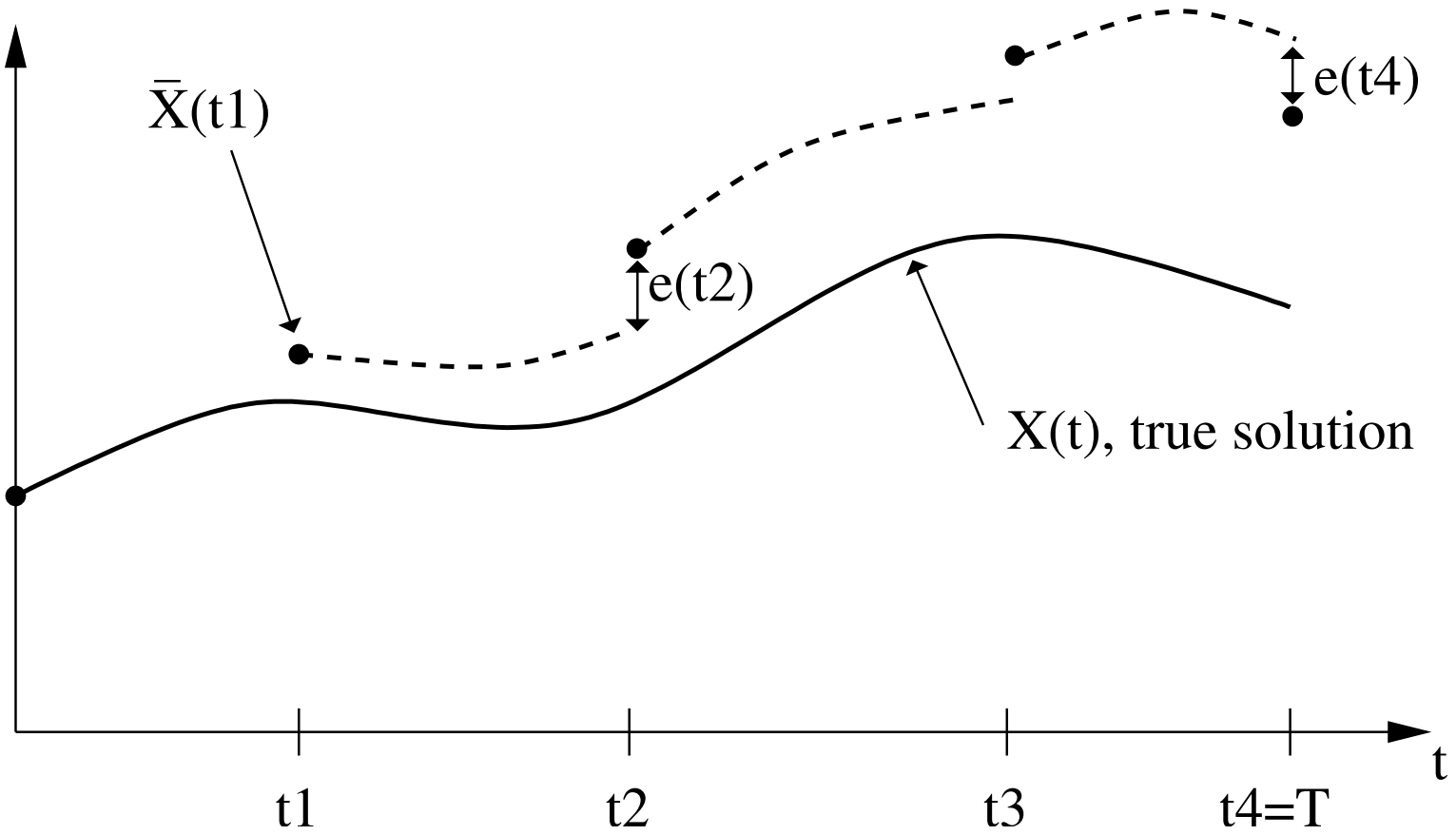
Definition (ψ)

Let X be the true solution of the Cauchy problem. Let $W \in \mathbb{R}^d$.
$\psi : \mathbb{R} \to \mathbb{R}^d$ measures the sensitivity of the goal function g around the true solution :

$$\big(\psi(s), W\big) = \lim_{\delta \to 0} \frac{u(s, X(s) + \delta W) - u(s, X(s))}{\delta}$$

ψ is the gradient of u with respect to the second variable.

Local Error



Local Error of an approximation

Definition (Local Solution)

Assume we approximate the true solution $X(t)$ by $\bar{X}(t)$ on a grid $0 = t_0 < t_1 < \ldots < t_N = T$. $\tilde{X}(t)$ is defined as follows, for $t_n < t \leq t_{n+1}$:

$$\begin{cases} \tilde{X}'(t) = a(t, \tilde{X}(t)) & \forall t \in (t_n, t_{n+1}] \\ \tilde{X}(t_n) = \bar{X}(t_n) \end{cases}$$

Definition (Local Error)

$e(t_n) = \tilde{X}(t_n) - \bar{X}(t_n)$.

Local Error of an approximation

Definition (Local Solution)

Assume we approximate the true solution $X(t)$ by $\bar{X}(t)$ on a grid $0 = t_0 < t_1 < \ldots < t_N = T$. $\tilde{X}(t)$ is defined as follows, for $t_n < t \leq t_{n+1}$:

$$\begin{cases} \tilde{X}'(t) = a(t, \tilde{X}(t)) & \forall t \in (t_n, t_{n+1}] \\ \tilde{X}(t_n) = \bar{X}(t_n) \end{cases}$$

Definition (Local Error)

$e(t_n) = \tilde{X}(t_n) - \bar{X}(t_n)$.

Central Theorem (intuitive)

Theorem

Assume that the Cauchy problem has a unique solution for all possible X_0. Assume that $a(t,x)$ is differentiable in x for all t in $[0\ T]$. For all differential functions g, the global error is a weighted sum of the local errors :

$$g(X(T)) - g(\bar{X}(T)) = \sum_{n=1}^{N} \left(e(t_n), \psi(t_n) \right)$$

and ψ satisfies

$$\begin{cases} -\dfrac{d\psi(s)}{ds} &= (a')^*(s, X(s))\ \psi(s) \\ \psi(T) &= \nabla g(X(T)) \end{cases}$$

The above problem is called the dual problem of the Cauchy problem.

Central Theorem (intuitive)

Theorem

Assume that the Cauchy problem has a unique solution for all possible X_0. Assume that $a(t,x)$ is differentiable in x for all t in $[0\ T]$. For all differential functions g, the global error is a weighted sum of the local errors :

$$g(X(T)) - g(\bar{X}(T)) = \sum_{n=1}^{N} \left(e(t_n), \psi(t_n) \right)$$

and ψ satisfies

$$\begin{cases} -\dfrac{d\psi(s)}{ds} &= (a')^*(s, X(s))\ \psi(s) \\ \psi(T) &= \nabla g(X(T)) \end{cases}$$

The above problem is called the dual problem of the Cauchy problem.

Central Theorem (intuitive)

Theorem

Assume that the Cauchy problem has a unique solution for all possible X_0. Assume that $a(t,x)$ is differentiable in x for all t in $[0\ T]$. For all differential functions g, the global error is a weighted sum of the local errors :

$$g(X(T)) - g(\bar{X}(T)) = \sum_{n=1}^{N} \Big(e(t_n), \psi(t_n) \Big)$$

and ψ satisfies

$$\begin{cases} -\dfrac{d\psi(s)}{ds} &= (a')^*(s, X(s))\ \psi(s) \\ \psi(T) &= \nabla g(X(T)) \end{cases}$$

The above problem is called the dual problem of the Cauchy problem.

Central Theorem (exact form)

Theorem

Assume that the Cauchy problem has a unique solution for all possible X_0. Assume that $a(t,x)$ is differentiable in x for all t in $[0\ T]$. For all differential functions g, the global error is a weighted sum of the local errors :

$$g(X(T)) - g(\bar{X}(T)) = \sum_{n=1}^{N} \left(e(t_n), \int_0^1 \psi(t_n, \bar{X}(t_n) + se(t_n))ds \right)$$

and ψ satisfies

$$\begin{cases} -\dfrac{d\psi(s,X(s))}{ds} &= (a')^*(s,X(s))\ \psi(s,X(s)) \\ \psi(T,X(T)) &= \nabla g(X(T)) \end{cases}$$

The above problem is called the dual problem of the Cauchy problem.

X and ψ, its dual

Primal

$$\left\{ \begin{array}{rcl} X'(t) & = & a(t, X(t)) \ \ \forall t \in [0\ T] \\ X(0) & = & X_0 \end{array} \right.$$

Dual

$$\left\{ \begin{array}{rcl} -\psi'(s) & = & (a')^*(s, X(s))\ \psi(s) \ \ \forall s \in [0\ T] \\ \psi(T) & = & \nabla g(X(T)) \end{array} \right.$$

Approximation of $e(t_n)$

To approximate $e(t_n) = \tilde{X}(t_n) - \bar{X}(t_n)$, we use another approximation $\bar{\bar{X}}(t_n)$ and we do Richardson extrapolation based on the different orders of the estimation $\bar{X}(t_n)$ and $\bar{\bar{X}}(t_n)$.

$$\bar{e}(t_n) = \gamma(\bar{\bar{X}}(t_n) - \bar{X}(t_n))$$

Approximation of $\psi(t_n)$

We replace the system

$$\left\{ \begin{array}{rcl} -\dfrac{d\psi(s)}{ds} & = & (a')^*(s, X(s))\ \psi(s) \\ \psi(T) & = & \nabla g(X(T)) \end{array} \right.$$

by the system

$$\left\{ \begin{array}{rcl} -\dfrac{d\psi(s)}{ds} & = & (a')^*(s, \bar{X}(s))\ \psi(s) \\ \psi(T) & = & \nabla g(\bar{X}(T)) \end{array} \right.$$

So, we can find ψ.

The algorithm

From the estimation of the global error

$$g(X(T)) - g(\bar{X}(T)) \approx \sum_{n=1}^{N} \underbrace{\left(\bar{e}(t_n), \bar{\psi}(t_n) \right)}_{r_n}$$

we will construct an adaptative algorithm to solve chaotic ordinary differential equations. The idea is to refine the mesh at the places where r_n is big.

The algorithm

Inputs

- TOL, the tolerance we want on $g(X(T)) - g(\bar{X}(T))$.
- A one-step numerical method M to approximate ODE (Euler progressive, Runge-Kutta, ...).
- N_0 the initial number of time steps.
- $a(\cdot, \cdot)$ the given function of the Cauchy Problem.
- $g(\cdot)$ the goal function.
- T the final time.
- X_0 the initial condition.

Outputs

- $\mathcal{T}$, the mesh the algorithm use to find the approximation.
- $\bar{X}$ the approximation of X on the mesh $\mathcal{T}$ that satisfies $g(X(T)) - g(\bar{X}(T)) < TOL$

The algorithm

Inputs

- TOL, the tolerance we want on $g(X(T)) - g(\bar{X}(T))$.
- A one-step numerical method M to approximate ODE (Euler progressive, Runge-Kutta, ...).
- N_0 the initial number of time steps.
- $a(\cdot,\cdot)$ the given function of the Cauchy Problem.
- $g(\cdot)$ the goal function.
- T the final time.
- X_0 the initial condition.

Outputs

- $\mathcal{T}$, the mesh the algorithm use to find the approximation.
- $\bar{X}$ the approximation of X on the mesh $\mathcal{T}$ that satisfies $g(X(T)) - g(\bar{X}(T)) < TOL$

The algorithm

Inputs

- TOL, the tolerance we want on $g(X(T)) - g(\bar{X}(T))$.
- A one-step numerical method M to approximate ODE (Euler progressive, Runge-Kutta, ...).
- N_0 the initial number of time steps.
- $a(\cdot, \cdot)$ the given function of the Cauchy Problem.
- $g(\cdot)$ the goal function.
- T the final time.
- X_0 the initial condition.

Outputs

- $\mathcal{T}$, the mesh the algorithm use to find the approximation.
- $\bar{X}$ the approximation of X on the mesh $\mathcal{T}$ that satisfies $g(X(T)) - g(\bar{X}(T)) < TOL$

The algorithm

Inputs

- TOL, the tolerance we want on $g(X(T)) - g(\bar{X}(T))$.
- A one-step numerical method M to approximate ODE (Euler progressive, Runge-Kutta, ...).
- N_0 the initial number of time steps.
- $a(\cdot, \cdot)$ the given function of the Cauchy Problem.
- $g(\cdot)$ the goal function.
- T the final time.
- X_0 the initial condition.

Outputs

- $\mathcal{T}$, the mesh the algorithm use to find the approximation.
- $\bar{X}$ the approximation of X on the mesh $\mathcal{T}$ that satisfies $g(X(T)) - g(\bar{X}(T)) < TOL$

The algorithm

Inputs

- TOL, the tolerance we want on $g(X(T)) - g(\bar{X}(T))$.
- A one-step numerical method M to approximate ODE (Euler progressive, Runge-Kutta, ...).
- N_0 the initial number of time steps.
- $a(\cdot, \cdot)$ the given function of the Cauchy Problem.
- $g(\cdot)$ the goal function.
- T the final time.
- X_0 the initial condition.

Outputs

- $\mathcal{T}$, the mesh the algorithm use to find the approximation.
- $\bar{X}$ the approximation of X on the mesh $\mathcal{T}$ that satisfies $g(X(T)) - g(\bar{X}(T)) < TOL$

The algorithm

1. $\mathcal{T}$ a uniform grid of N_0 timesteps.
2. Compute $\bar{X}$ on the mesh $\mathcal{T}$ using the numerical method M.
3. Compute $\bar{\bar{X}}$ on a different mesh using M.
4. Use extrapolation to compute the approximation of the local error $\bar{e} = \gamma(\bar{\bar{X}} - \bar{X})$.
5. Compute the weights ψ on the mesh $\mathcal{T}$ using M.
6. Compute the residuals $r_i = (e_i, \psi_i)$
7. Compute the estimation of the error $E = \sum_i r_i$. If $E < TOL$ stop.
8. Refine the mesh where r_i is big.
9. Go back to 2.

We call this algorithm `mstz`.

The algorithm

1. $\mathcal{T}$ a uniform grid of N_0 timesteps.
2. Compute $\bar{X}$ on the mesh $\mathcal{T}$ using the numerical method M.
3. Compute $\bar{\bar{X}}$ on a different mesh using M.
4. Use extrapolation to compute the approximation of the local error $\bar{e} = \gamma(\bar{\bar{X}} - \bar{X})$.
5. Compute the weights ψ on the mesh $\mathcal{T}$ using M.
6. Compute the residuals $r_i = (e_i, \psi_i)$
7. Compute the estimation of the error $E = \sum_i r_i$. If $E < TOL$ stop.
8. Refine the mesh where r_i is big.
9. Go back to 2.

We call this algorithm `mstz`.

The algorithm

1. $\mathcal{T}$ a uniform grid of N_0 timesteps.
2. Compute $\bar{X}$ on the mesh $\mathcal{T}$ using the numerical method M.
3. Compute $\bar{\bar{X}}$ on a different mesh using M.
4. Use extrapolation to compute the approximation of the local error $\bar{e} = \gamma(\bar{\bar{X}} - \bar{X})$.
5. Compute the weights ψ on the mesh $\mathcal{T}$ using M.
6. Compute the residuals $r_i = (e_i, \psi_i)$
7. Compute the estimation of the error $E = \sum_i r_i$. If $E < TOL$ stop.
8. Refine the mesh where r_i is big.
9. Go back to 2.

We call this algorithm `mstz`.

The algorithm

1. $\mathcal{T}$ a uniform grid of N_0 timesteps.
2. Compute $\bar{X}$ on the mesh $\mathcal{T}$ using the numerical method M.
3. Compute $\bar{\bar{X}}$ on a different mesh using M.
4. Use extrapolation to compute the approximation of the local error $\bar{e} = \gamma(\bar{\bar{X}} - \bar{X})$.
5. Compute the weights ψ on the mesh $\mathcal{T}$ using M.
6. Compute the residuals $r_i = (e_i, \psi_i)$
7. Compute the estimation of the error $E = \sum_i r_i$. If $E < TOL$ stop.
8. Refine the mesh where r_i is big.
9. Go back to 2.

We call this algorithm `mstz`.

The algorithm

1. $\mathcal{T}$ a uniform grid of N_0 timesteps.
2. Compute $\bar{X}$ on the mesh $\mathcal{T}$ using the numerical method M.
3. Compute $\bar{\bar{X}}$ on a different mesh using M.
4. Use extrapolation to compute the approximation of the local error $\bar{e} = \gamma(\bar{\bar{X}} - \bar{X})$.
5. Compute the weights ψ on the mesh $\mathcal{T}$ using M.
6. Compute the residuals $r_i = (e_i, \psi_i)$
7. Compute the estimation of the error $E = \sum_i r_i$. If $E < TOL$ stop.
8. Refine the mesh where r_i is big.
9. Go back to 2.

We call this algorithm `mstz`.

The algorithm

1. $\mathcal{T}$ a uniform grid of N_0 timesteps.
2. Compute $\bar{X}$ on the mesh $\mathcal{T}$ using the numerical method M.
3. Compute $\bar{\bar{X}}$ on a different mesh using M.
4. Use extrapolation to compute the approximation of the local error $\bar{e} = \gamma(\bar{\bar{X}} - \bar{X})$.
5. Compute the weights ψ on the mesh $\mathcal{T}$ using M.
6. Compute the residuals $r_i = (e_i, \psi_i)$
7. Compute the estimation of the error $E = \sum_i r_i$. If $E < TOL$ stop.
8. Refine the mesh where r_i is big.
9. Go back to 2.

We call this algorithm `mstz`.

The algorithm

1. $\mathcal{T}$ a uniform grid of N_0 timesteps.
2. Compute $\bar{X}$ on the mesh $\mathcal{T}$ using the numerical method M.
3. Compute $\bar{\bar{X}}$ on a different mesh using M.
4. Use extrapolation to compute the approximation of the local error $\bar{e} = \gamma(\bar{\bar{X}} - \bar{X})$.
5. Compute the weights ψ on the mesh $\mathcal{T}$ using M.
6. Compute the residuals $r_i = (e_i, \psi_i)$
7. Compute the estimation of the error $E = \sum_i r_i$. If $E < TOL$ stop.
8. Refine the mesh where r_i is big.
9. Go back to 2.

We call this algorithm `mstz`.

The algorithm

1. $\mathcal{T}$ a uniform grid of N_0 timesteps.
2. Compute $\bar{X}$ on the mesh $\mathcal{T}$ using the numerical method M.
3. Compute $\bar{\bar{X}}$ on a different mesh using M.
4. Use extrapolation to compute the approximation of the local error $\bar{e} = \gamma(\bar{\bar{X}} - \bar{X})$.
5. Compute the weights ψ on the mesh $\mathcal{T}$ using M.
6. Compute the residuals $r_i = (e_i, \psi_i)$
7. Compute the estimation of the error $E = \sum_i r_i$. If $E < TOL$ stop.
8. Refine the mesh where r_i is big.
9. Go back to 2.

We call this algorithm `mstz`.

The algorithm

1. $\mathcal{T}$ a uniform grid of N_0 timesteps.
2. Compute $\bar{X}$ on the mesh $\mathcal{T}$ using the numerical method M.
3. Compute $\bar{\bar{X}}$ on a different mesh using M.
4. Use extrapolation to compute the approximation of the local error $\bar{e} = \gamma(\bar{\bar{X}} - \bar{X})$.
5. Compute the weights ψ on the mesh $\mathcal{T}$ using M.
6. Compute the residuals $r_i = (e_i, \psi_i)$
7. Compute the estimation of the error $E = \sum_i r_i$. If $E < TOL$ stop.
8. Refine the mesh where r_i is big.
9. Go back to 2.

We call this algorithm `mstz`.

Adaptive Runge-Kutta

- One of the most used method for solving ODE. Based on the Dormand-Prince pair. Use a pair of Runge-Kutta methods of order 4 and 5 to estimate the local error.
- A typical example of the strategy A.
- Implemented in MATLAB under the name `ode45`.
- Need to give to the solver the tolerance ε we want on the local error.

As there is no estimation of the global error with `ode45`, we decided to give to the solver the exact solution $X(T)$ and we decrease ε until we have $g(X(T)) - g(\bar{X}(T))$.

Adaptive Runge-Kutta and global error

1. Set the tolerance on the local error $\varepsilon = \dfrac{\mathrm{TOL}}{N_0}$.
2. Compute $\bar{X}$ using ode45.
3. Compute

$$E = g(X(T)) - g(\bar{X}(T))$$

If $E < TOL$, stop. Else set $\varepsilon = \varepsilon/10$ and go back to 2

We call this algorithm RK45.

Adaptive Runge-Kutta and global error

1. Set the tolerance on the local error $\varepsilon = \dfrac{\text{TOL}}{N_0}$.
2. Compute $\bar{X}$ using `ode45`.
3. Compute

$$E = g(X(T)) - g(\bar{X}(T))$$

If $E < TOL$, stop. Else set $\varepsilon = \varepsilon/10$ and go back to 2

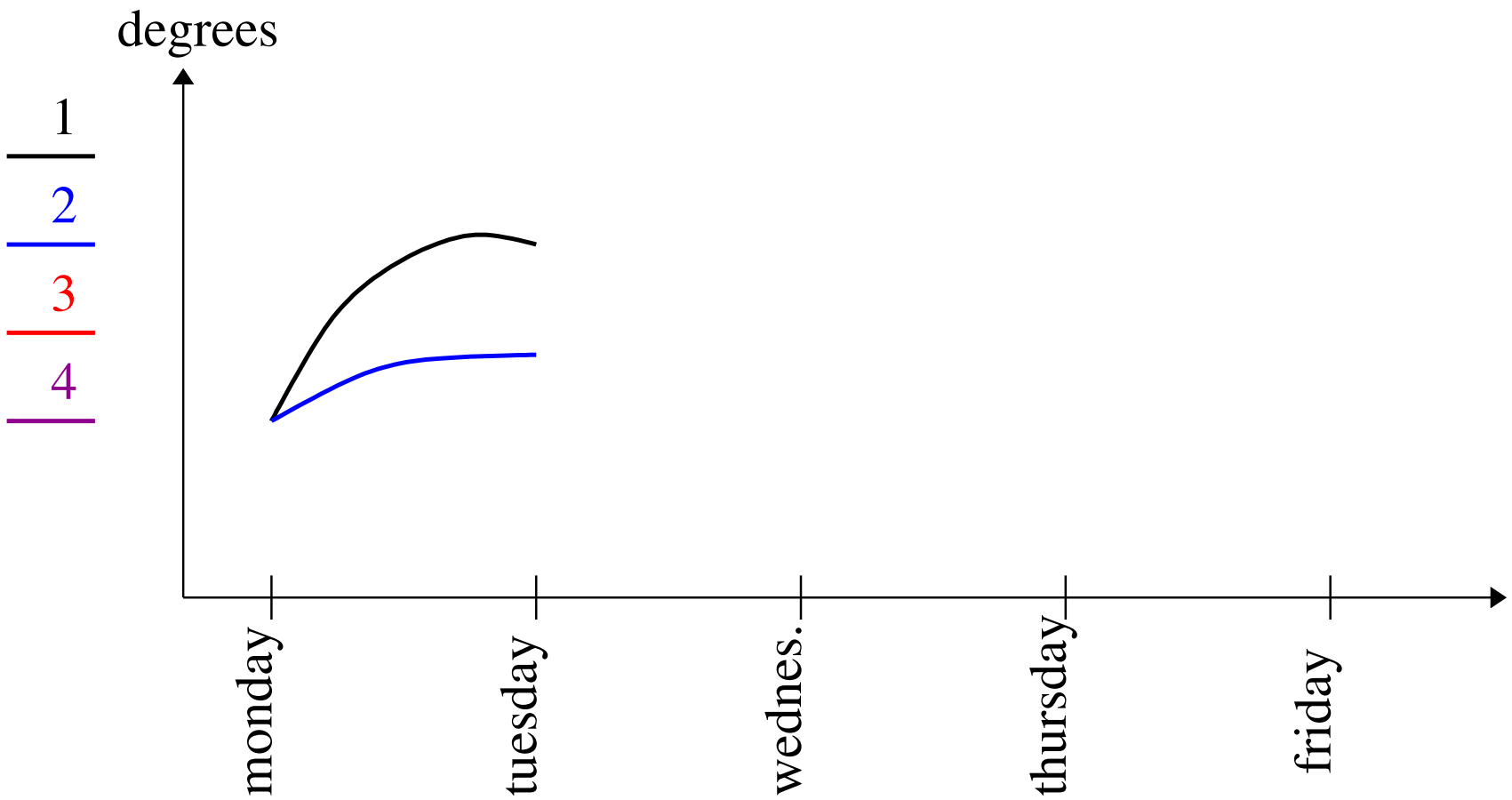
We call this algorithm `RK45`.

Adaptive Runge-Kutta and global error

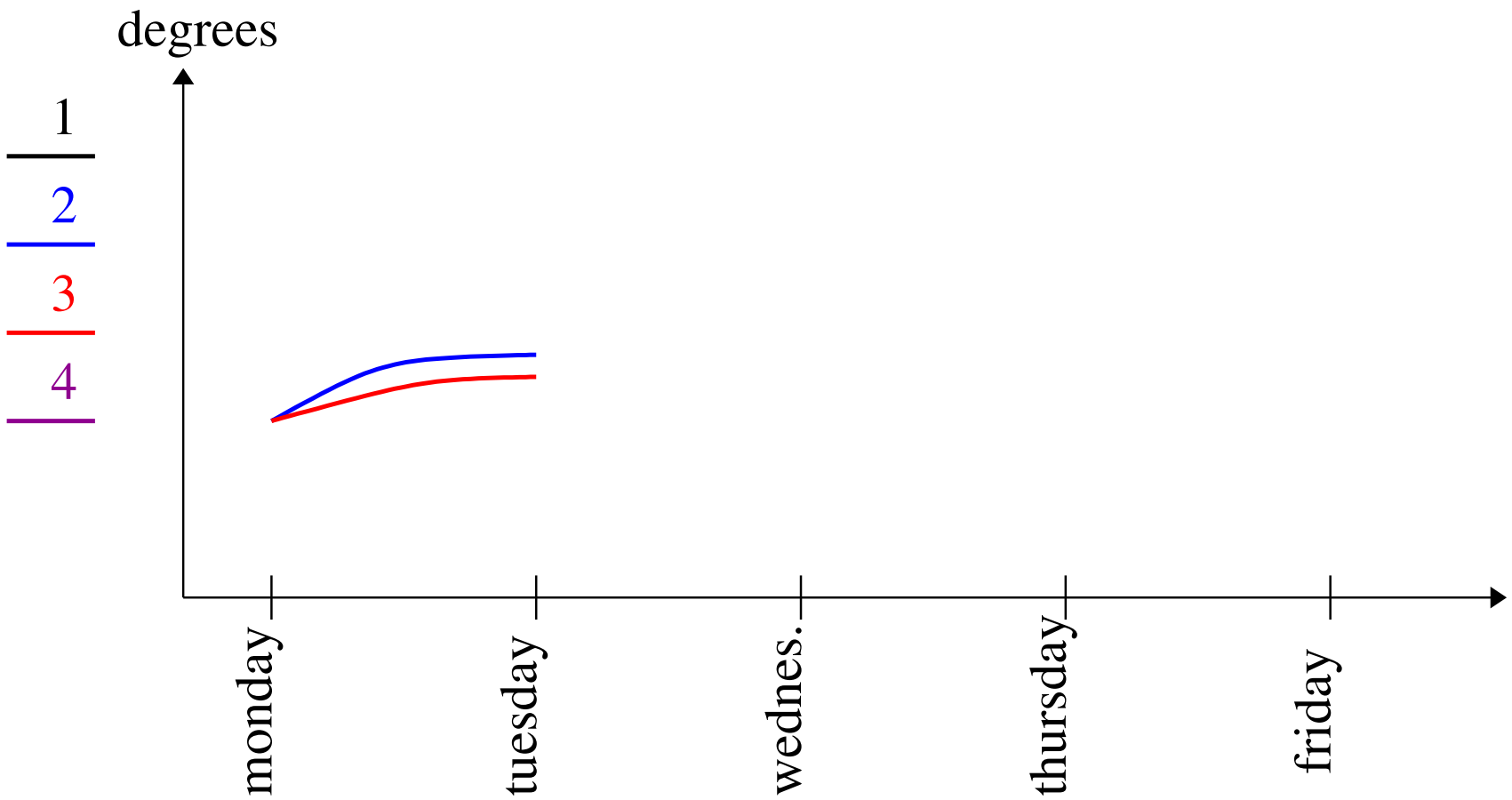
1. Set the tolerance on the local error $\varepsilon = \dfrac{\text{TOL}}{N_0}$.
2. Compute $\bar{X}$ using `ode45`.
3. Compute
$$E = g(X(T)) - g(\bar{X}(T))$$
If $E < TOL$, stop. Else set $\varepsilon = \varepsilon/10$ and go back to 2

We call this algorithm `RK45`.

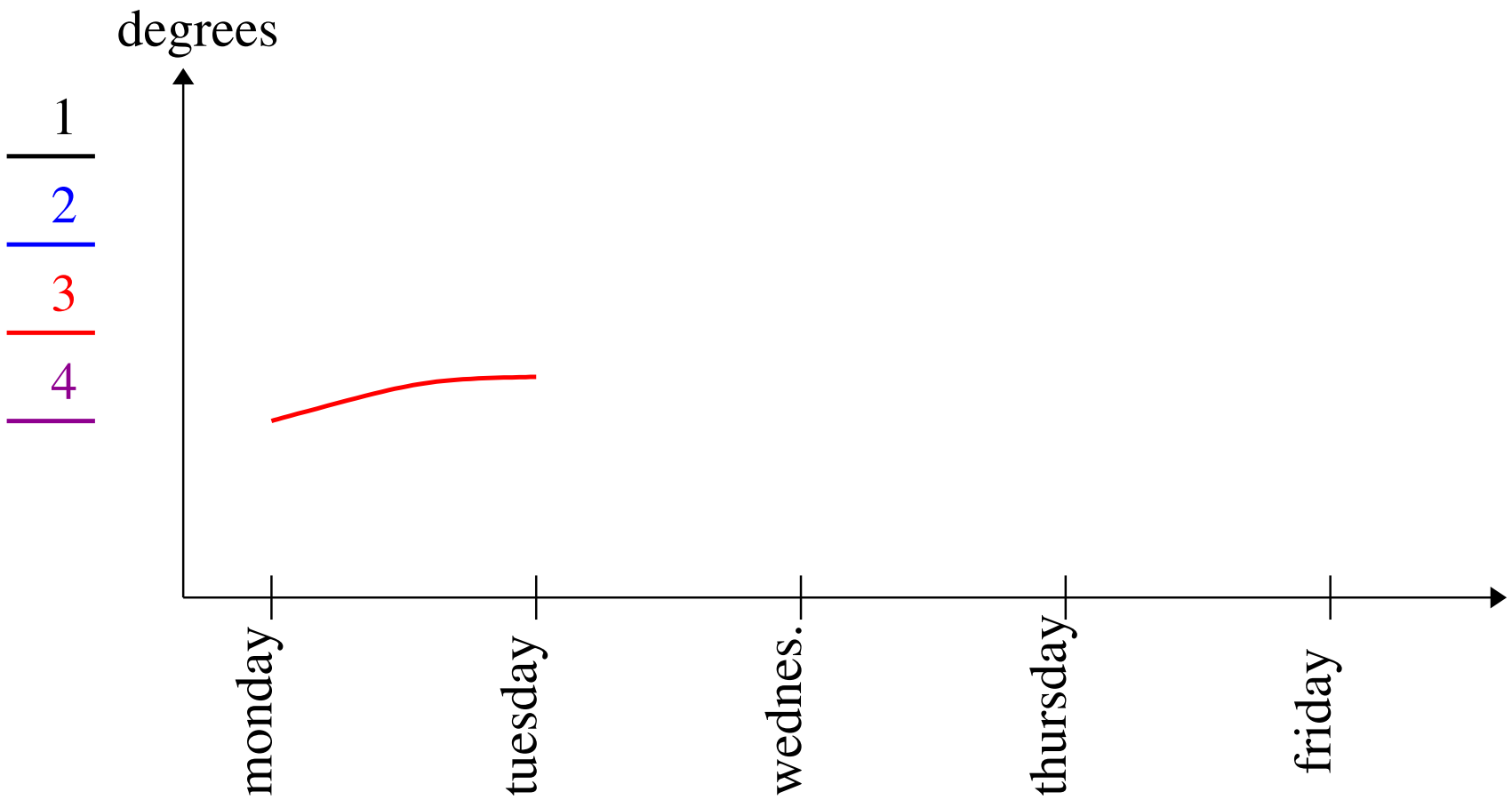
Strategy A



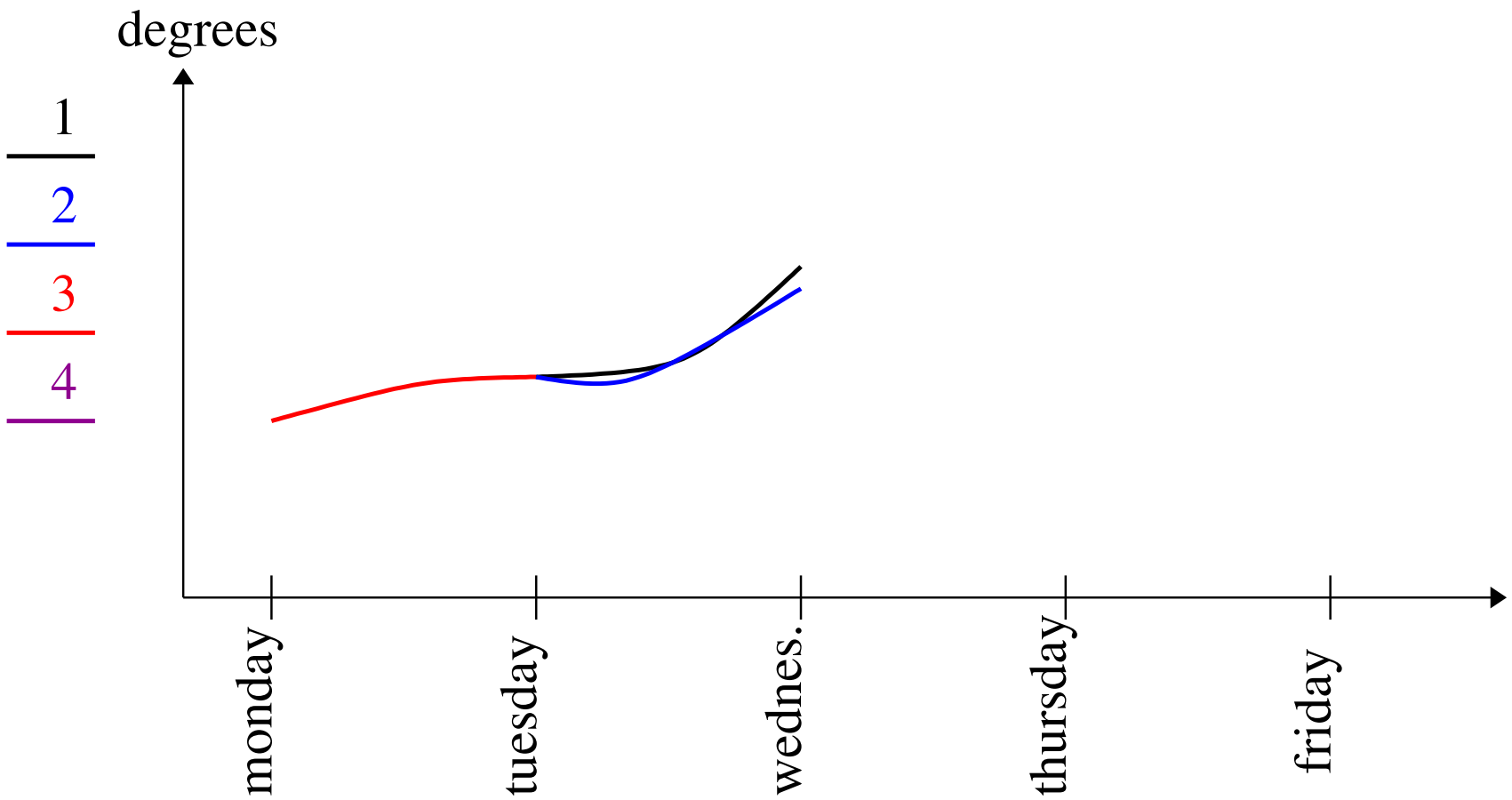
Strategy A



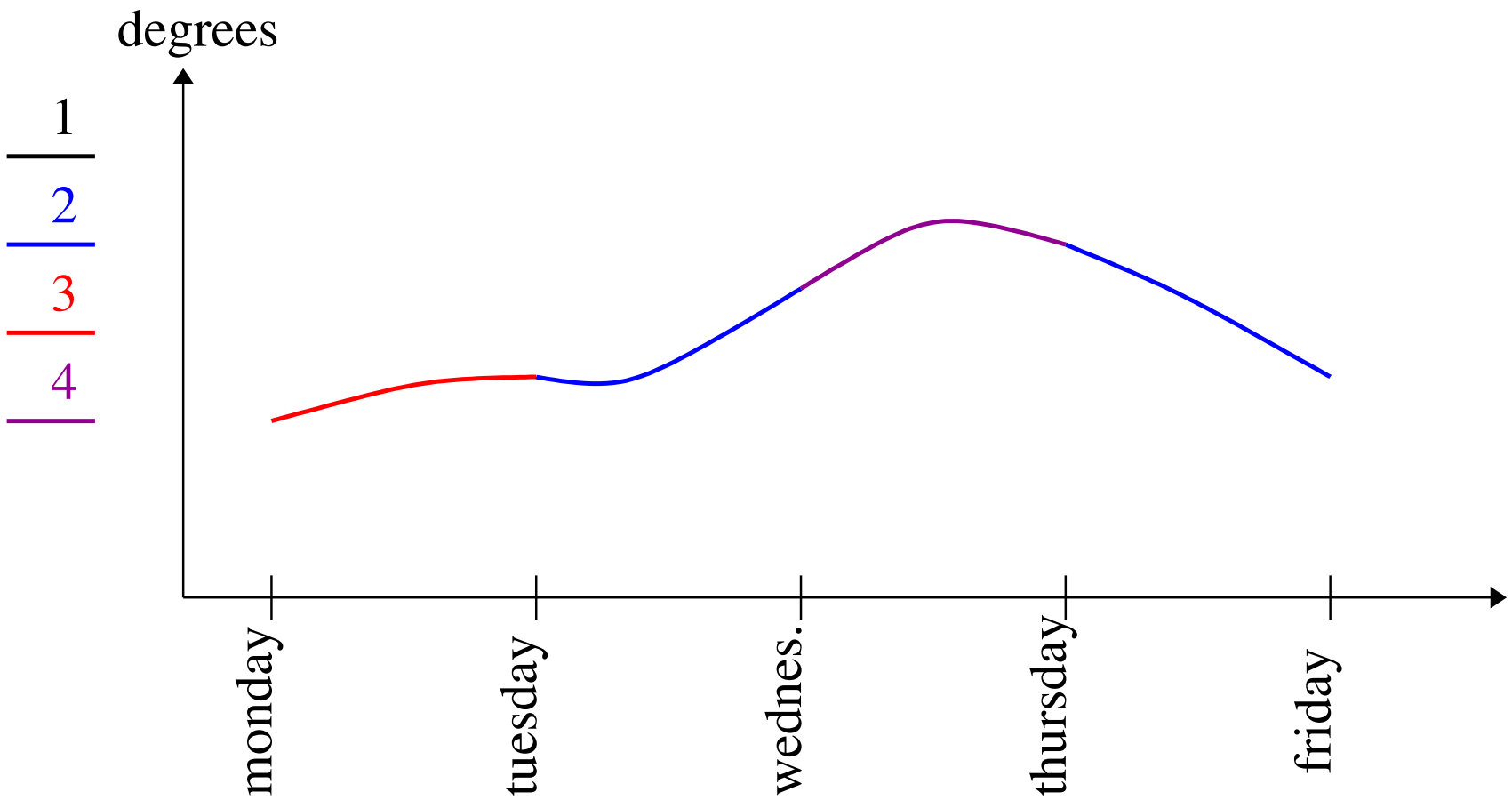
Strategy A



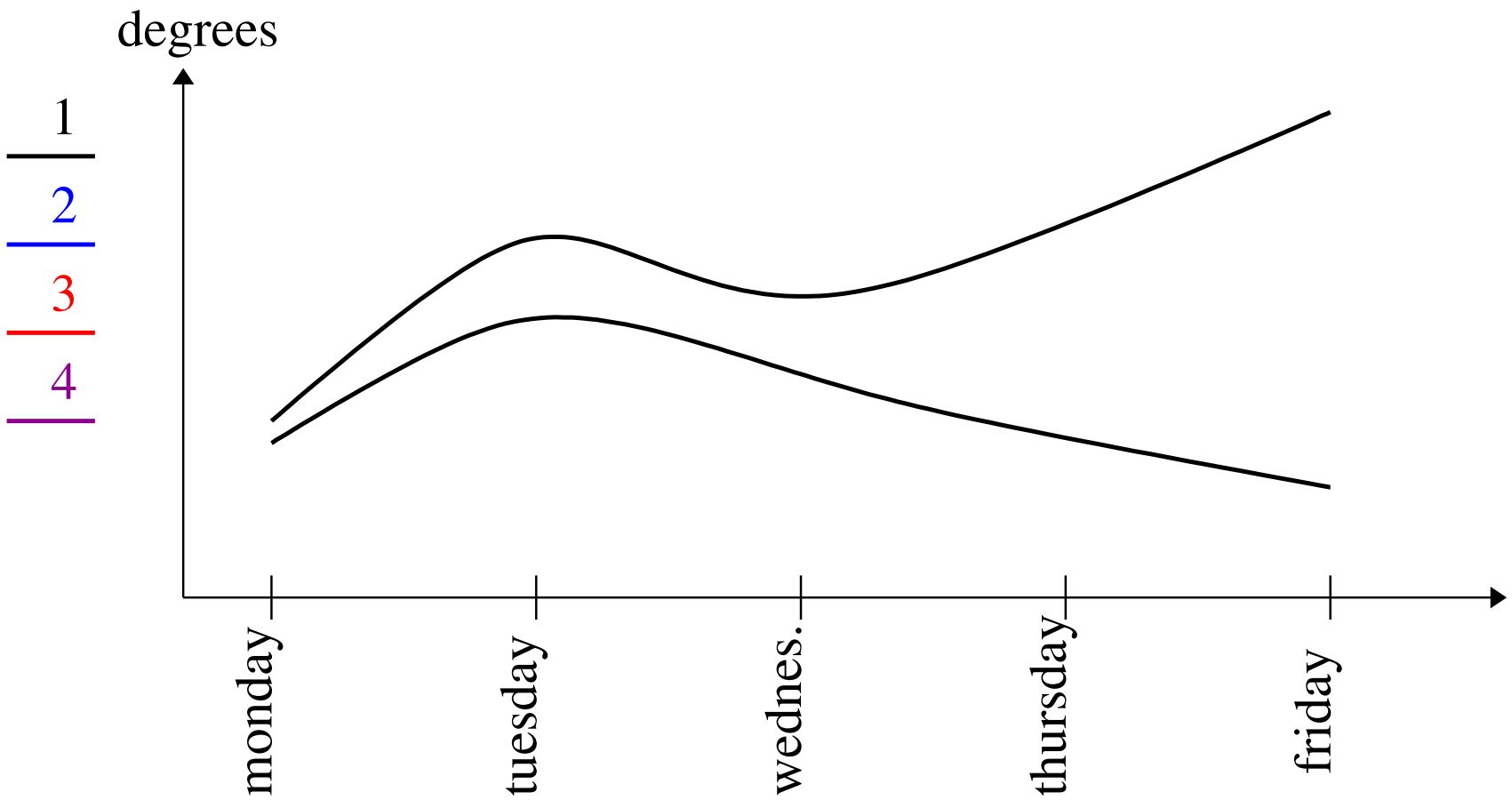
Strategy A



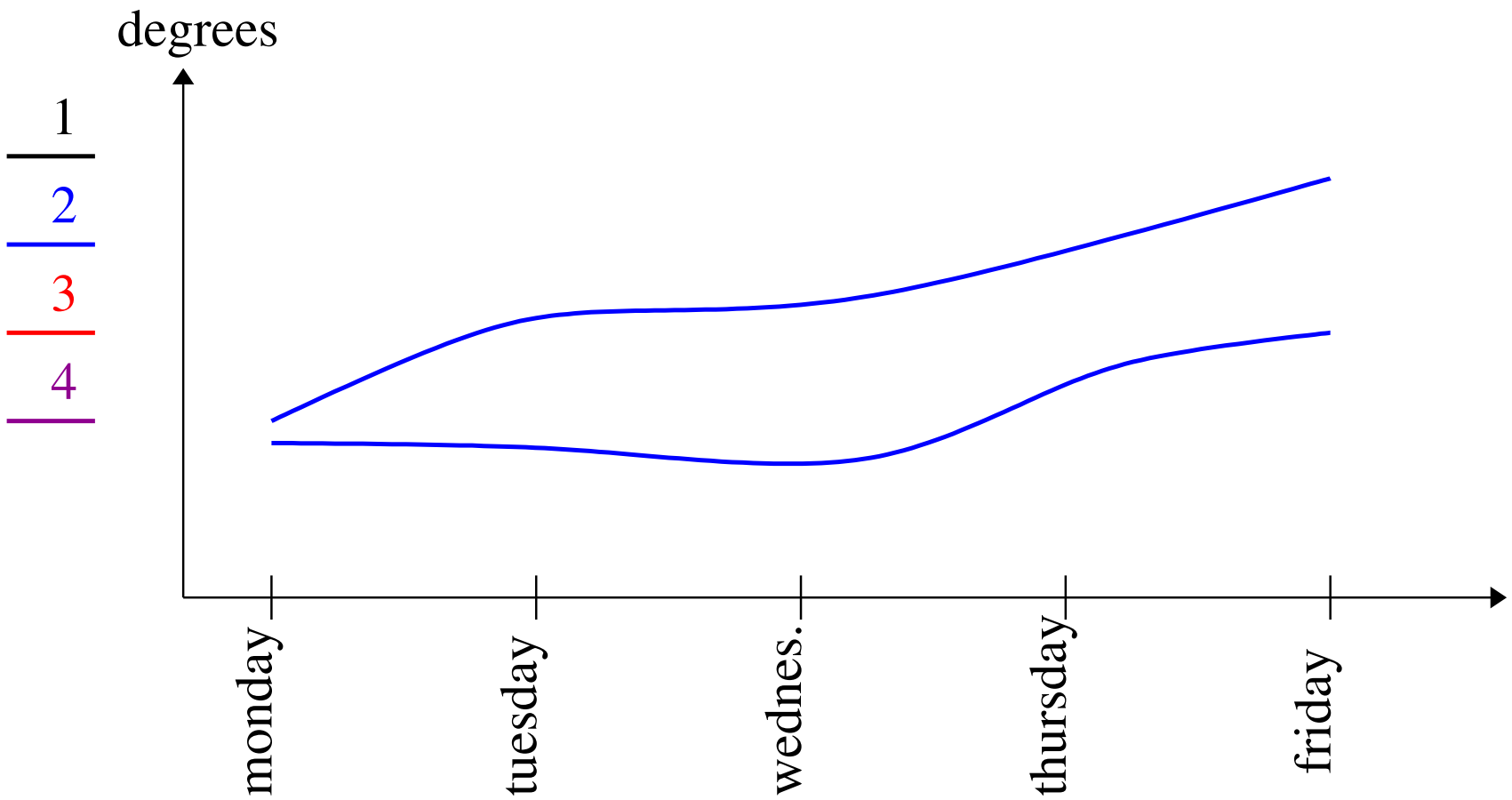
Strategy A



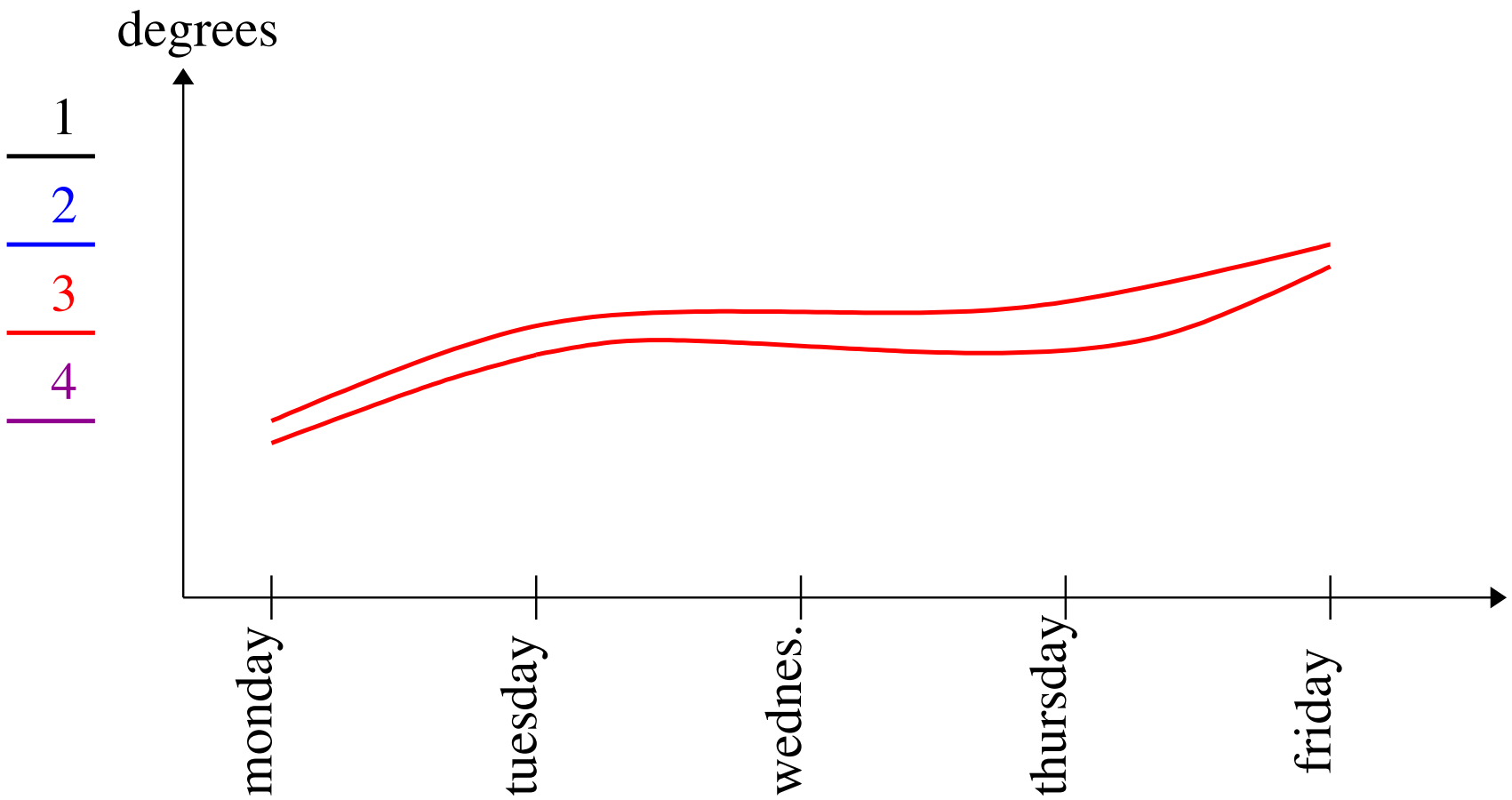
A better strategy : Strategy B



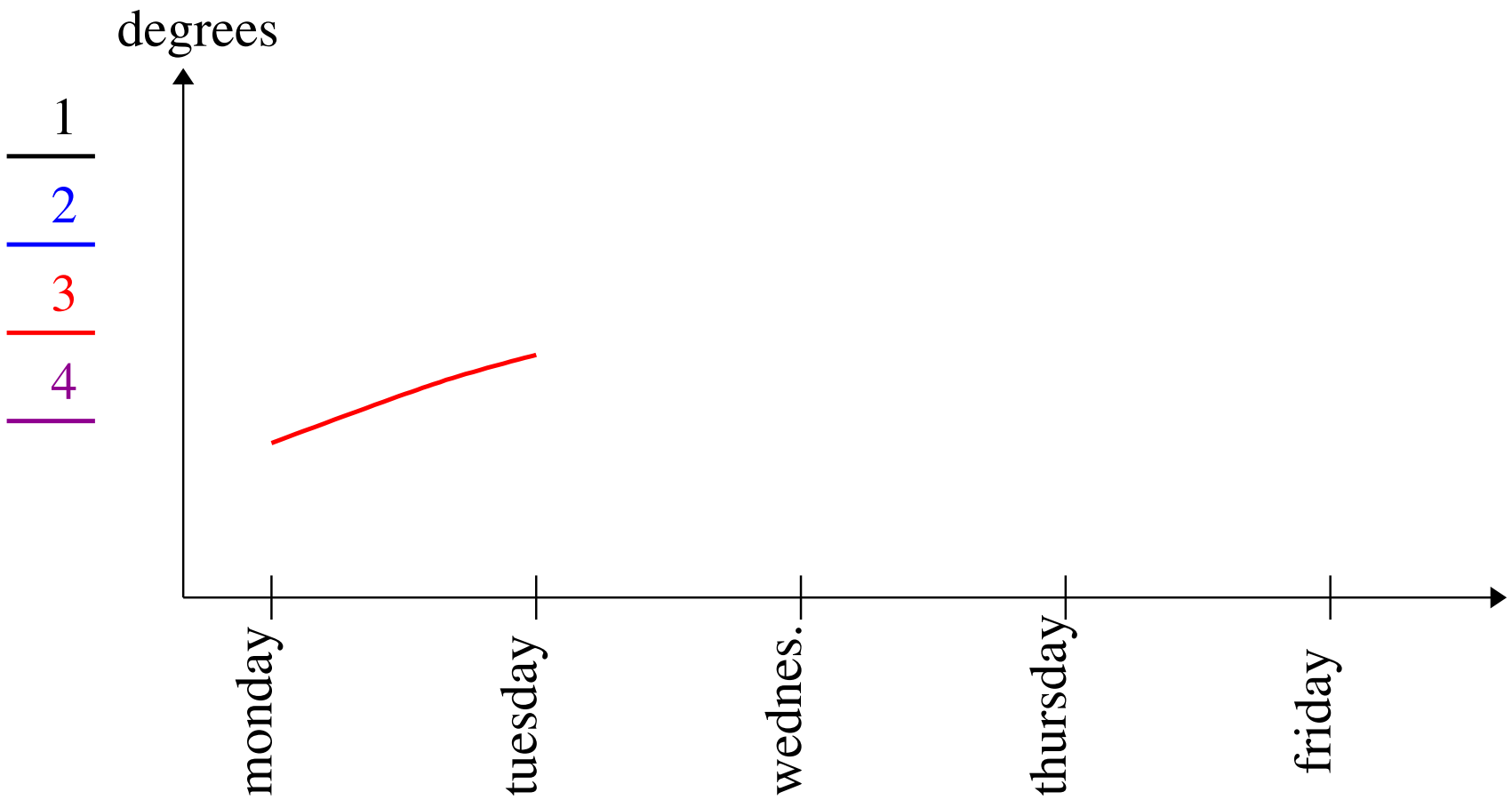
A better strategy : Strategy B



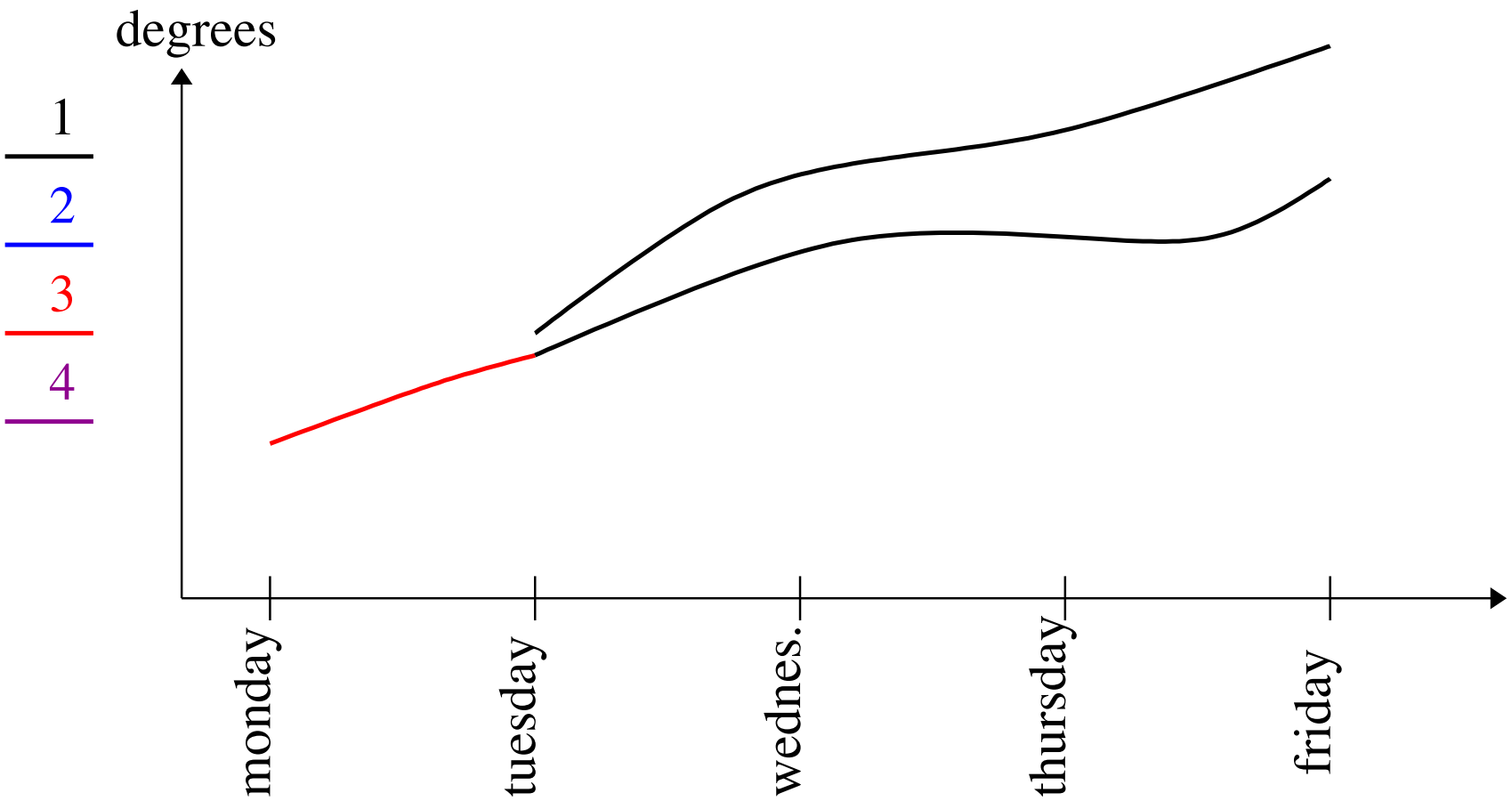
A better strategy : Strategy B



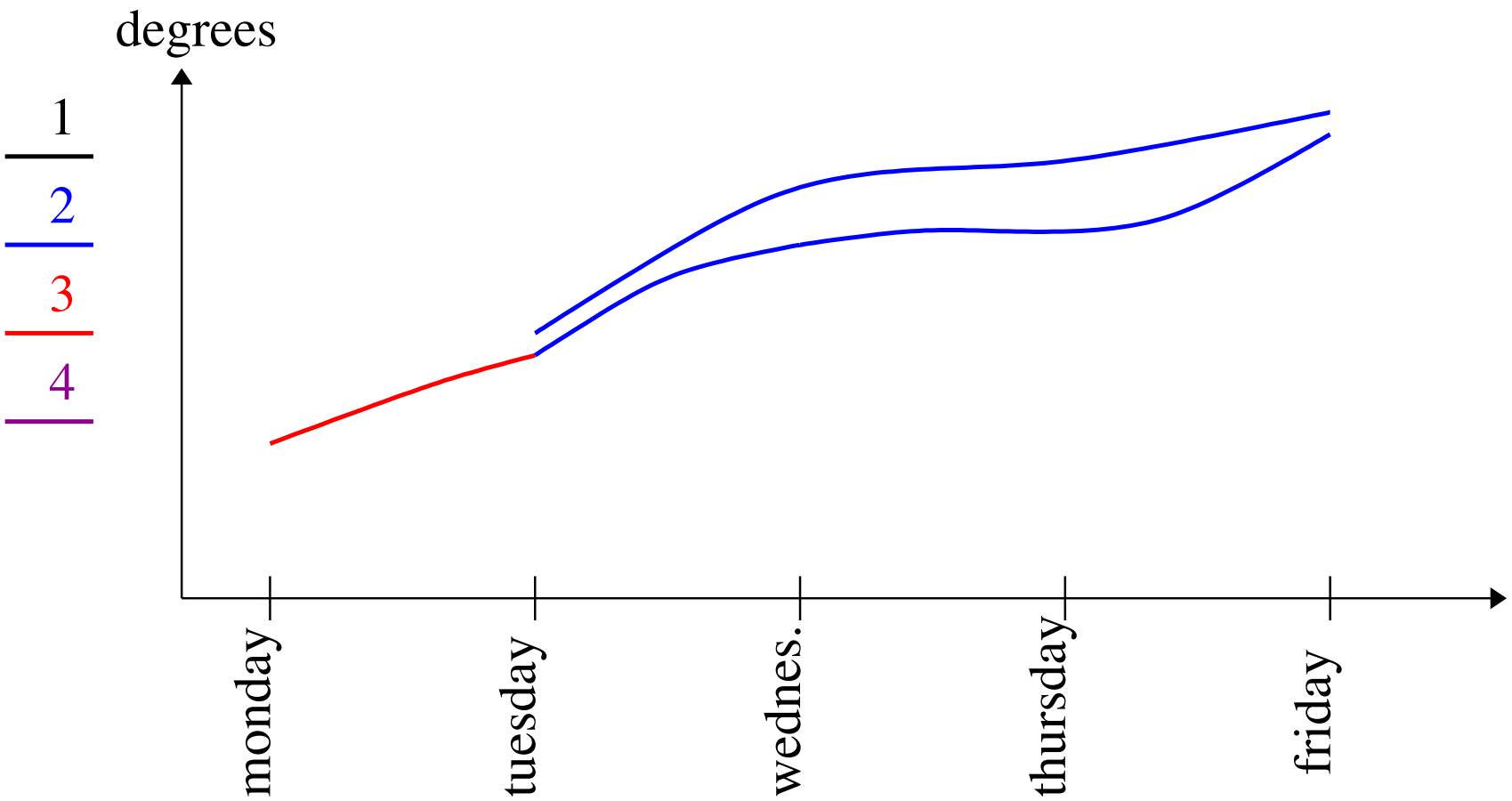
A better strategy : Strategy B



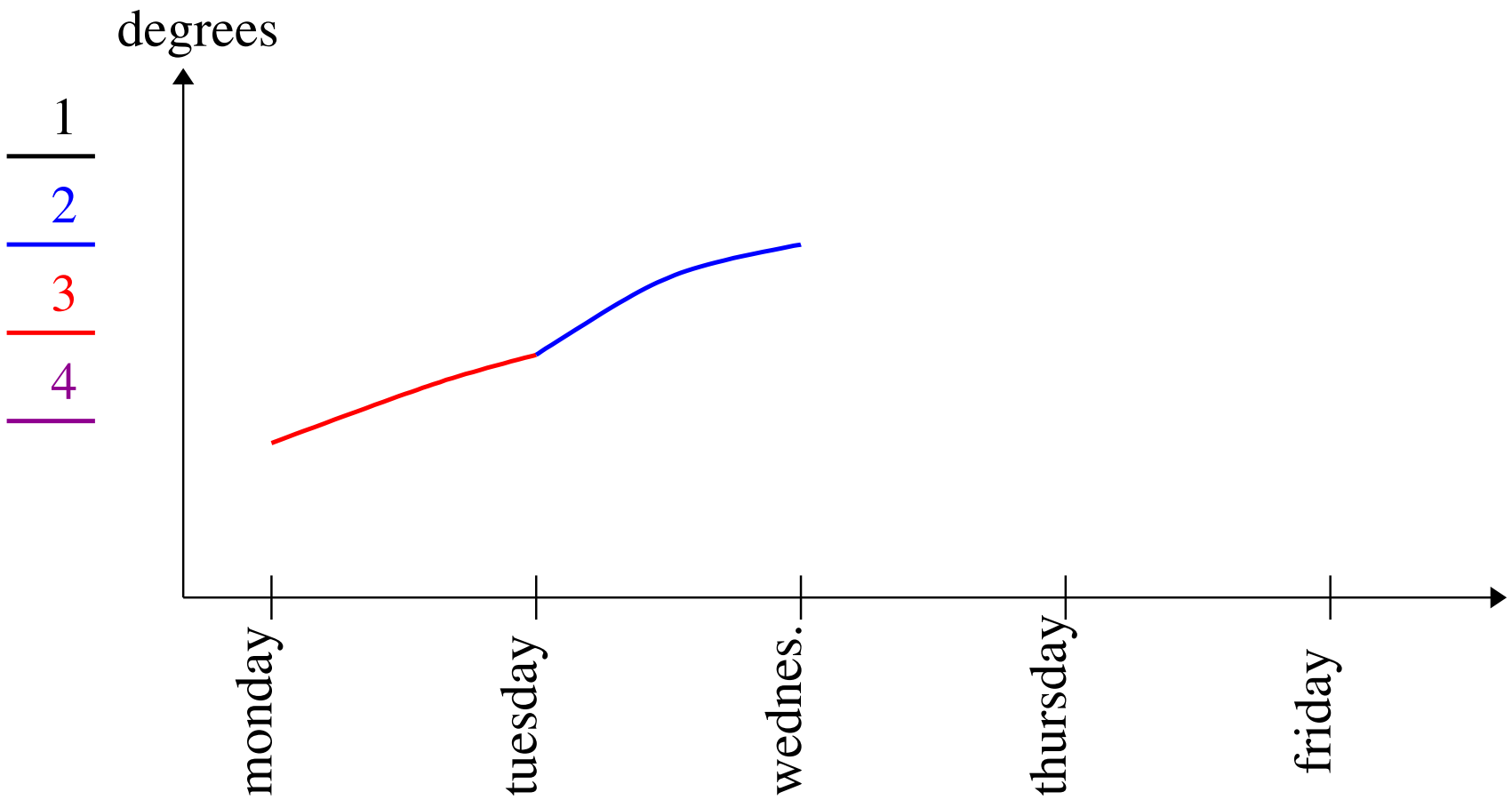
A better strategy : Strategy B



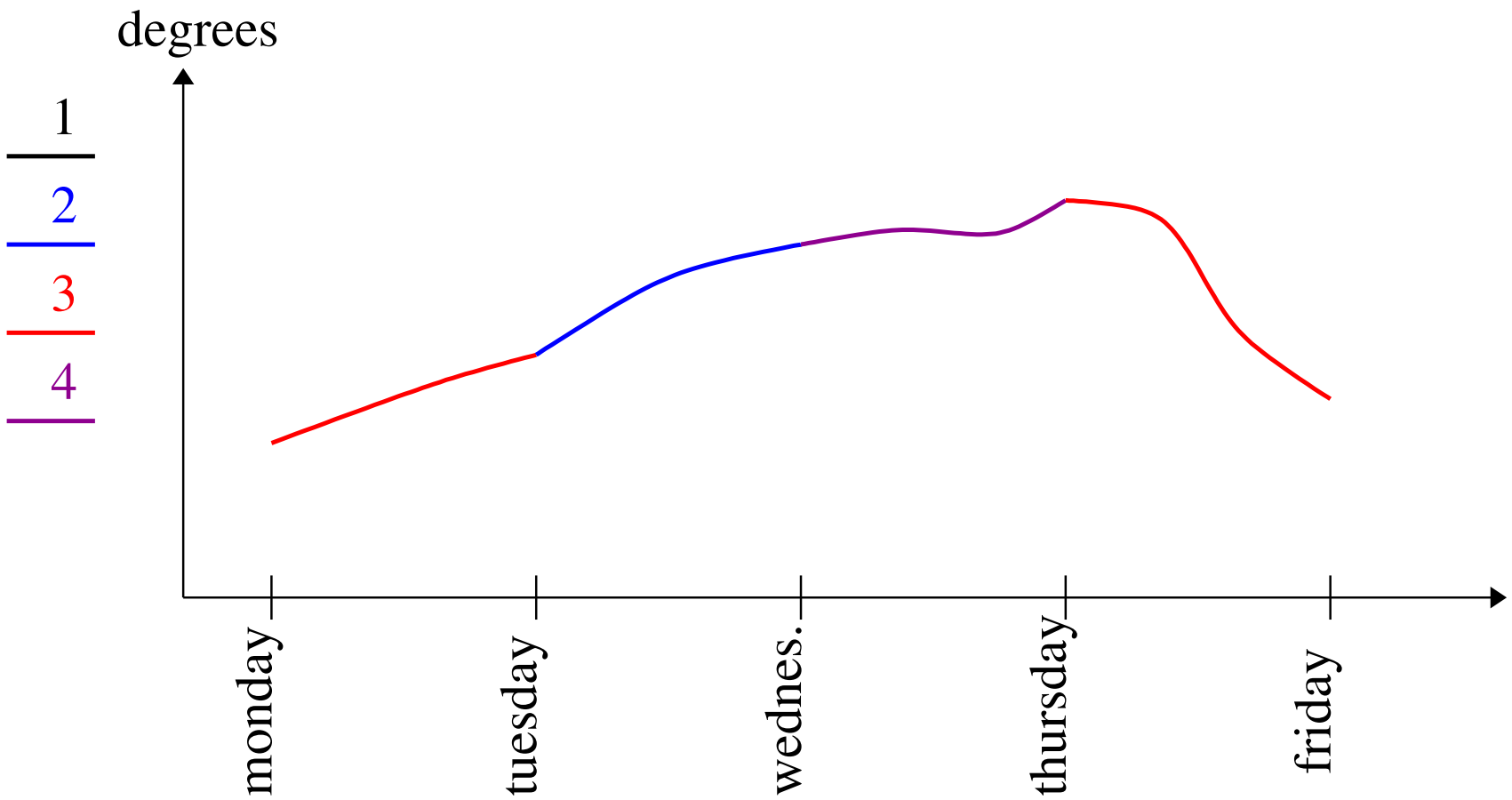
A better strategy : Strategy B



A better strategy : Strategy B



A better strategy : Strategy B



Test Cases

We will have 6 test cases :

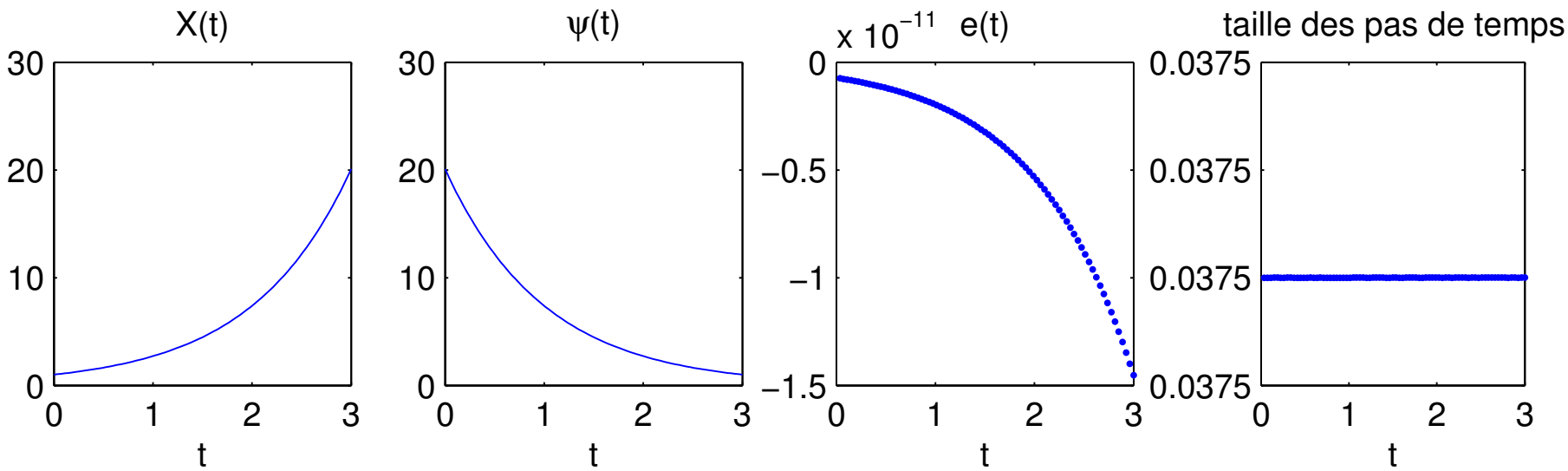
- The Exponential
- A non-linear problem with blow-up
- A stiff problem
- A problem with a discontinuity in the derivative
- A model of the transition to the turbulence
- The Lorenz problem

The last 2 problems are chaotic. For the numerical method we use a Runge-Kutta method of order 5. The coefficients are taken from the Dormand-Prince pair.

The Exponential

$$(a)\begin{cases} X'(t) &= X(t)\ \forall t \in [0\ 3] \\ X(0) &= 1 \\ g(x) &= x \\ \text{TOL} &= 10^{-8} \\ N_0 &= 5 \end{cases}$$

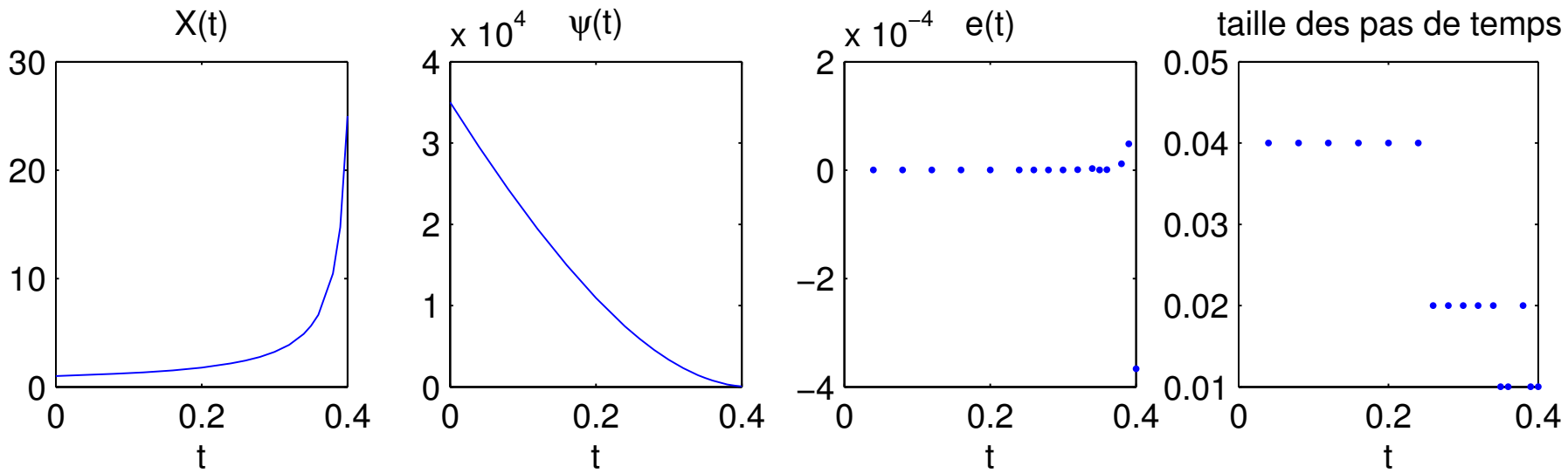
solution : $X(t) = \mathrm{e}^t$



Non-linear with blow-up

$$(b)\begin{cases} X'(t) &= 2(t+1)X^2(t) \quad \forall t \in [0\ 0.4] \\ X(0) &= 1 \\ g(x) &= x^2 \\ \text{TOL} &= 0.1 \\ N_0 &= 5 \end{cases}$$

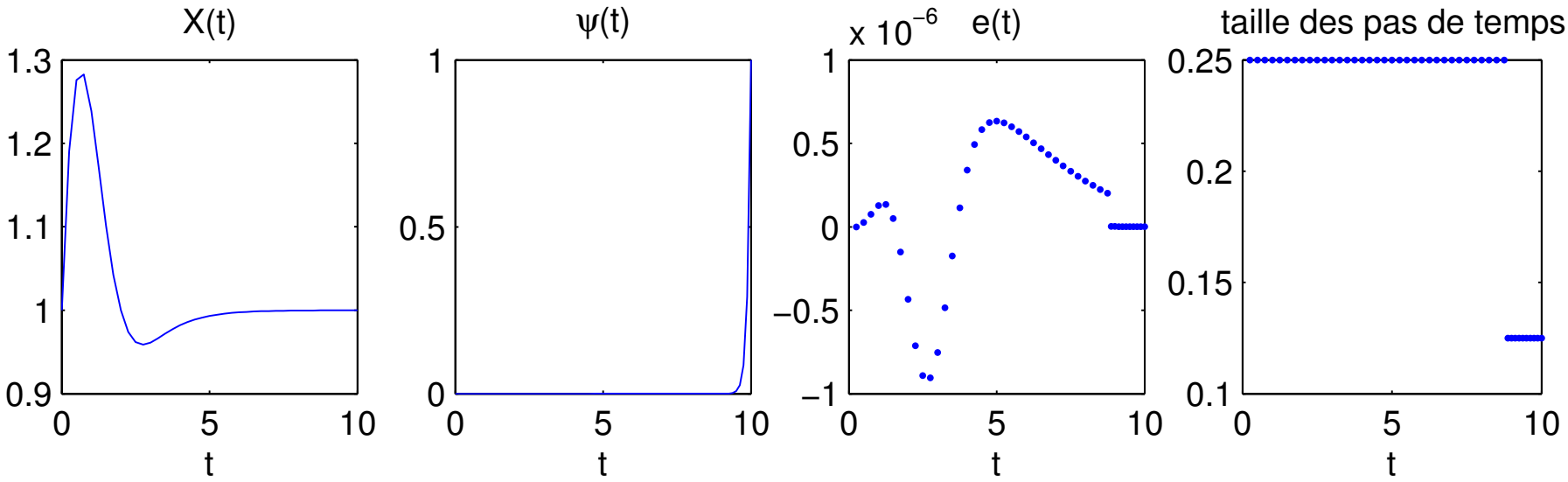
solution : $X(t) = \dfrac{-1}{t^2 + 2t - 1}$



Stiff problem

$$(c)\begin{cases} X'(t) &= t(1-X(t))+(1-t)\mathrm{e}^{-t}\ \forall t \in [0\ 10] \\ X(0) &= 1 \\ g(x) &= x \\ \mathrm{TOL} &= 10^{-8} \\ N_0 &= 5 \end{cases}$$

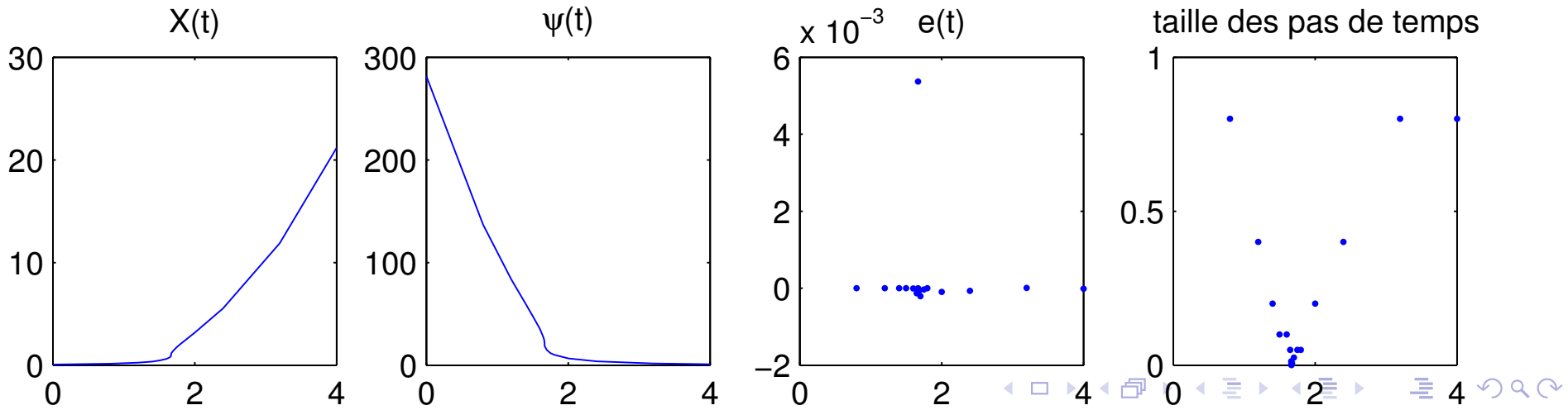
solution : $X(t) = \mathrm{e}^{-t^2/2} - \mathrm{e}^{-t} + 1.$



Singularity

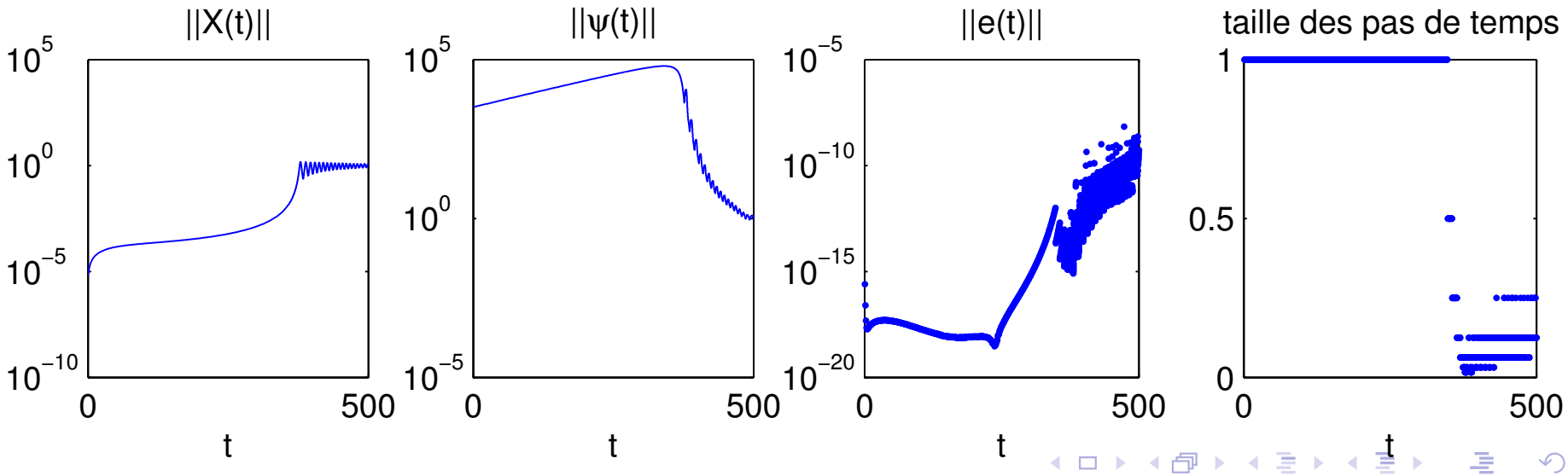
$$(d)\begin{cases} X'(t) &= \dfrac{X(t)}{\sqrt{t-5/3+\pi 10^{-8}}} \quad \forall t \in [0\ 10] \\ X(0) &= \mathrm{e}^{-2\sqrt{\frac{5}{3}-\pi 10^{-8}}} \\ g(x) &= x \\ \mathrm{TOL} &= 0.1 \\ N_0 &= 5 \end{cases}$$

$$\text{sol. : } X(t) = \exp(2 \cdot \operatorname{sign}(t-5/3+\pi 10^{-8}) \cdot \sqrt{|t-5/3+\pi 10^{-8}|})$$

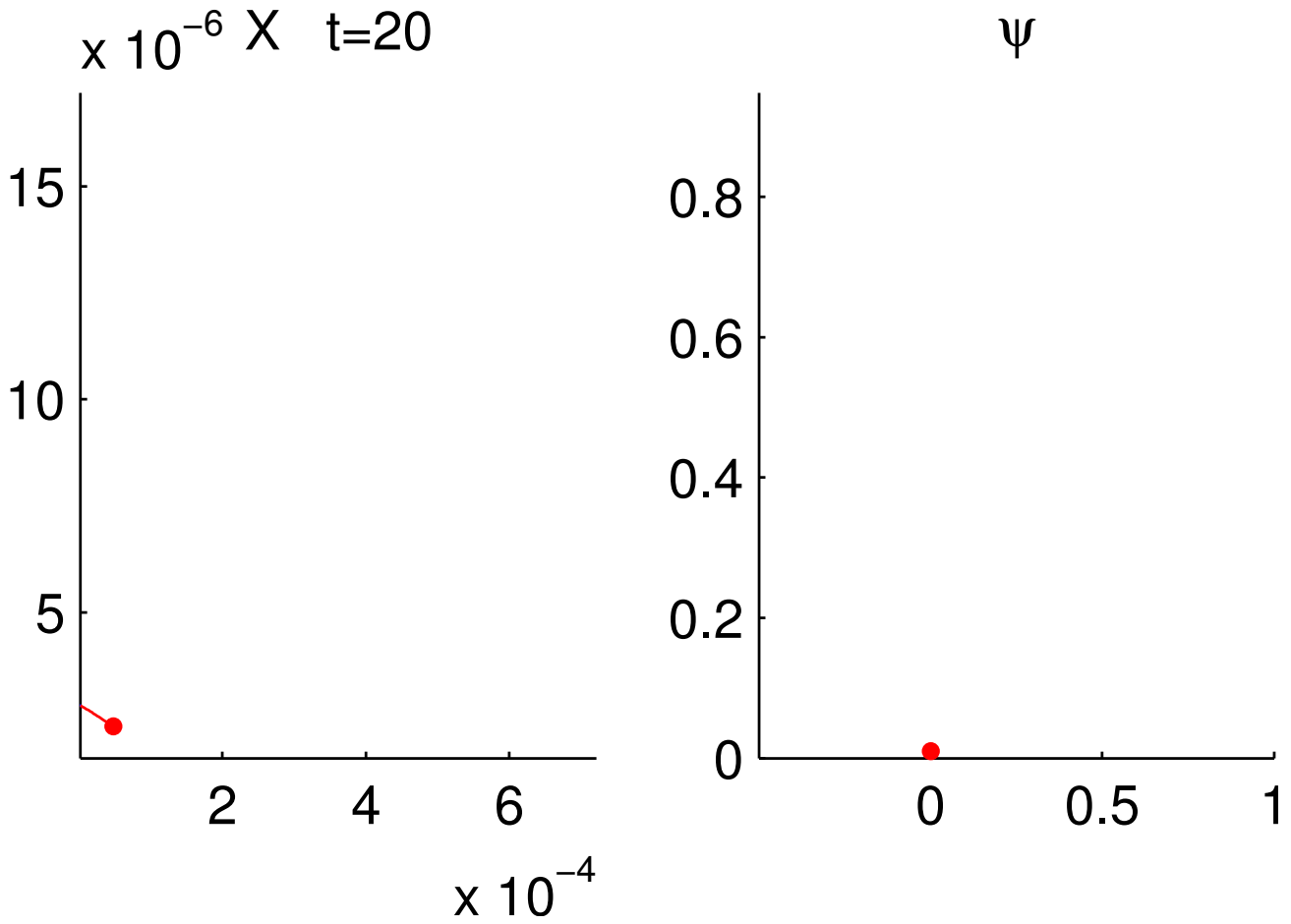


Transition to turbulence

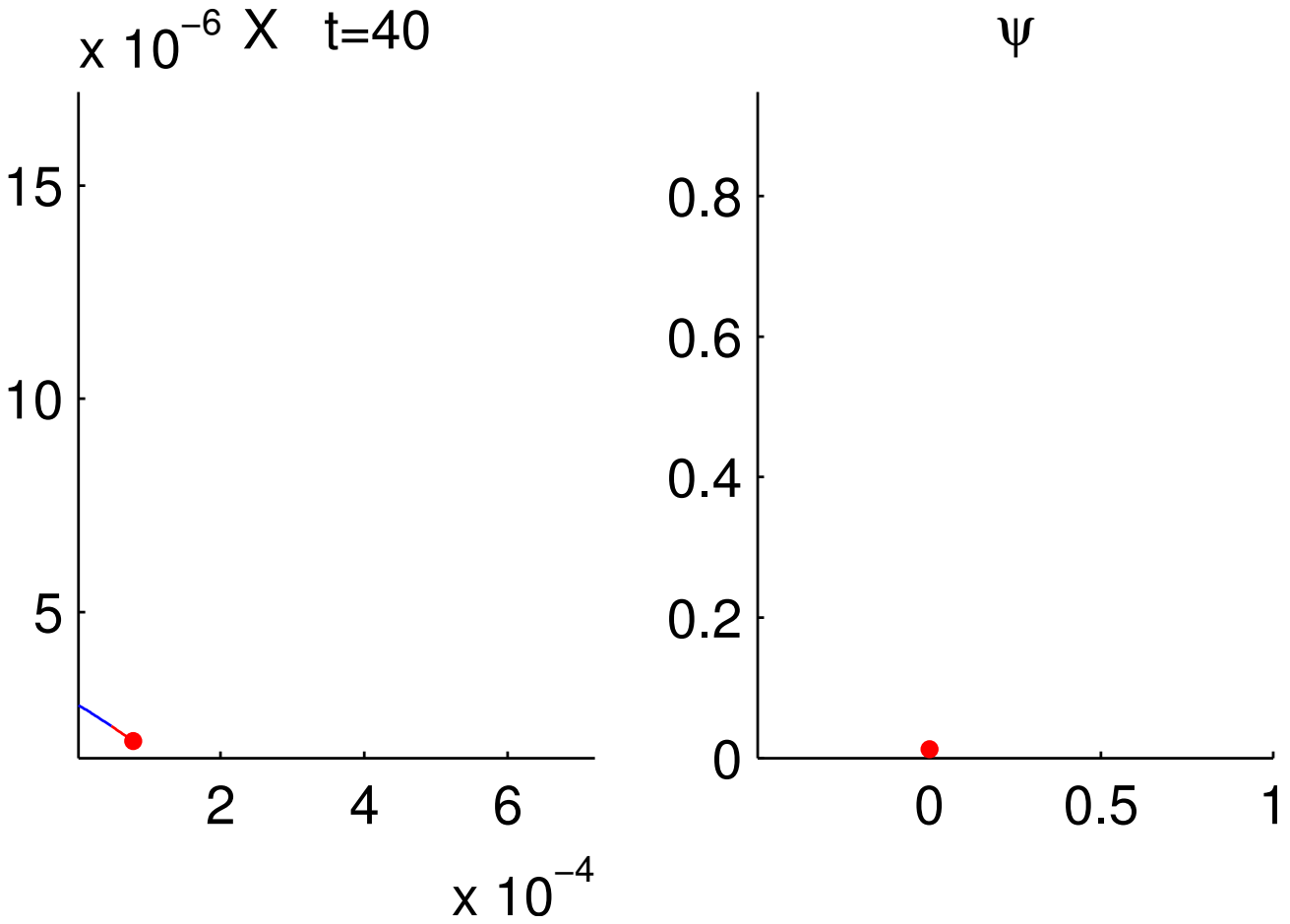
$$(t)\begin{cases} X'(t) &= \begin{pmatrix} -R^{-1} & 1 \\ 0 & -R^{-1} \end{pmatrix} X(t) + ||X(t)|| \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} X(t) \\ X(0) &= \dfrac{\delta}{\sqrt{2}}(1,1) \quad \delta > 0 \\ g(X) &= x_1 \text{ (the first component)} \\ \text{TOL} &= 10^{-6} \\ N_0 &= 500 \end{cases}$$



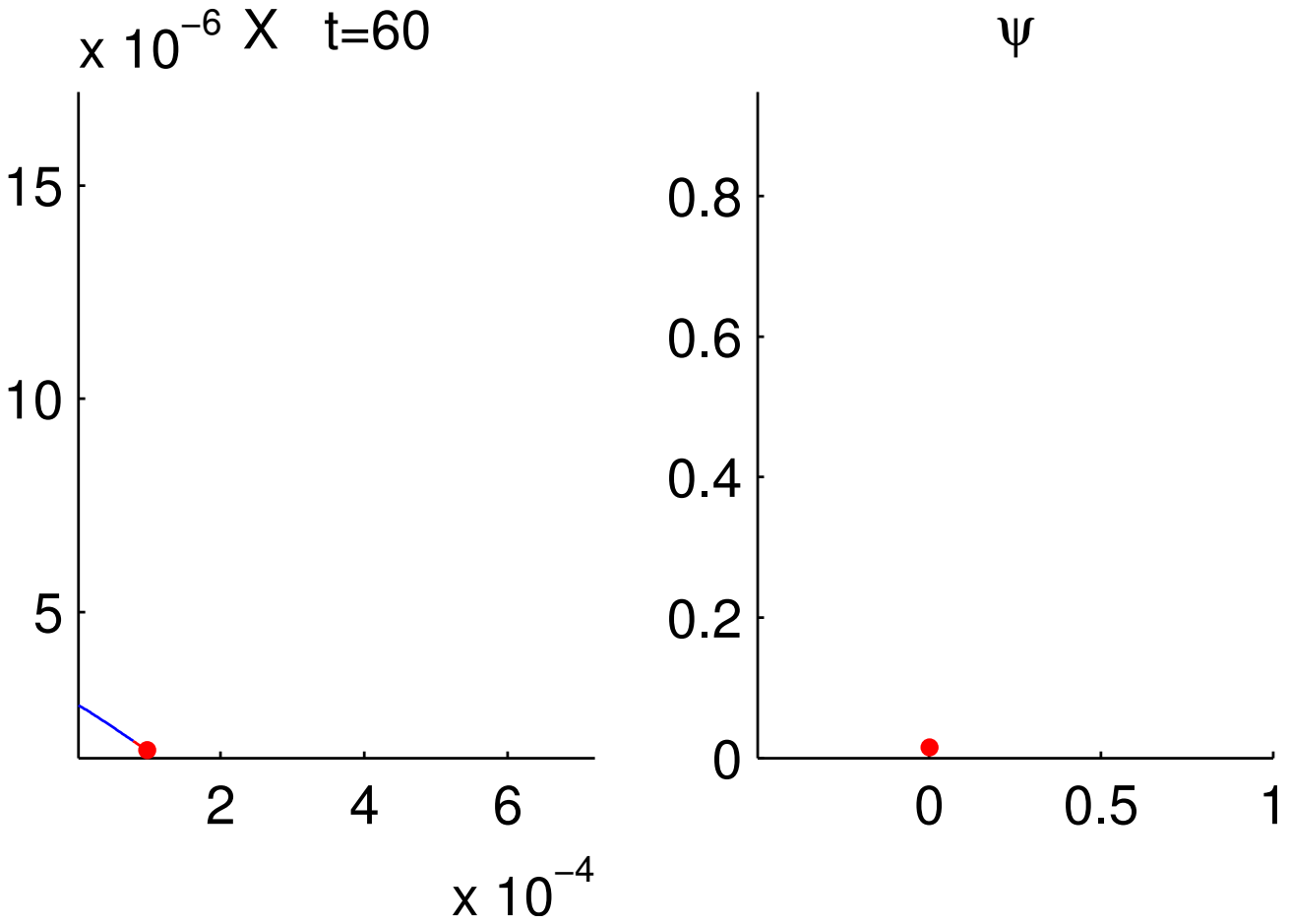
Transition to turbulence $\|X_0\| = 10^{-5.4}$



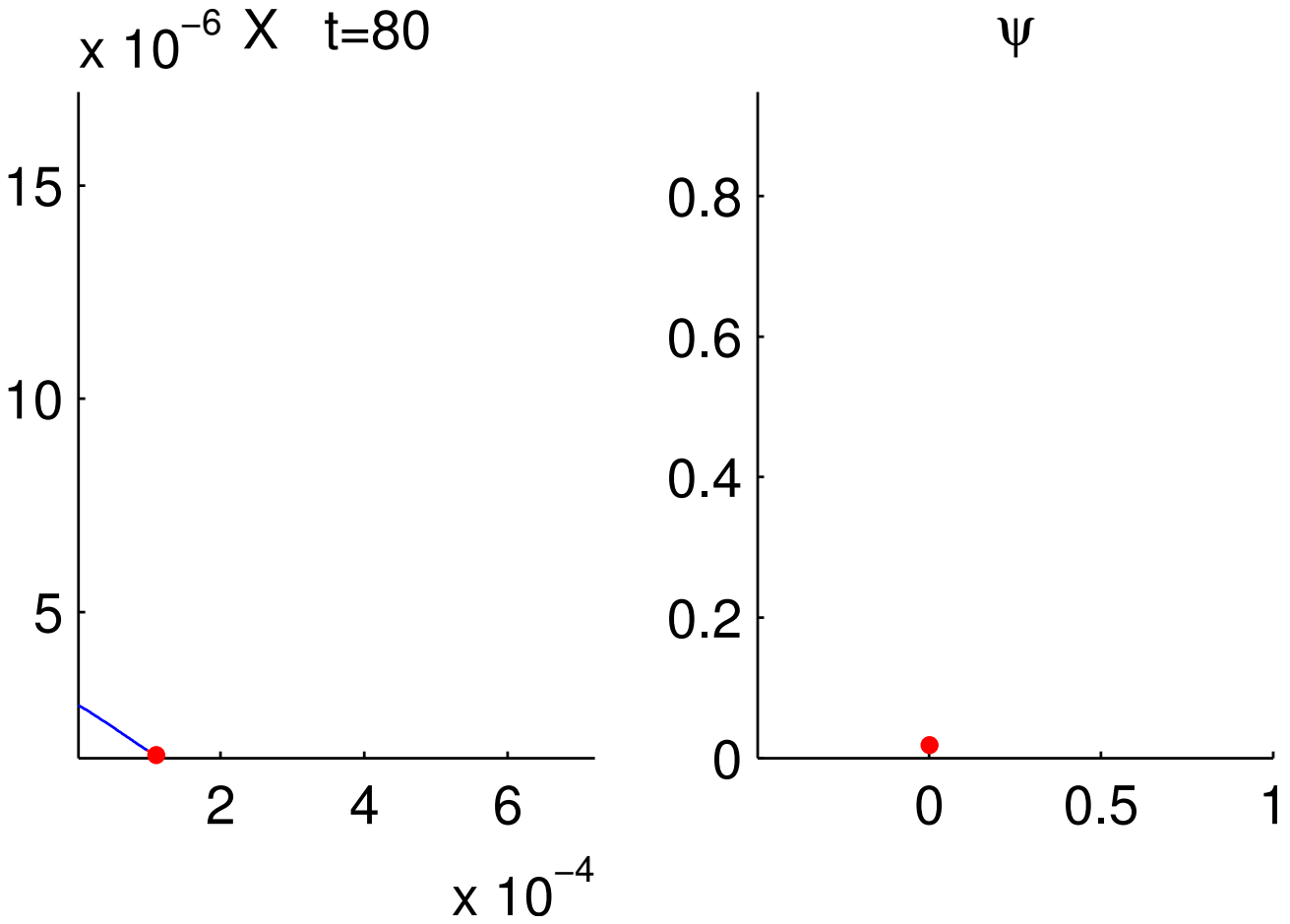
Transition to turbulence $\|X_0\| = 10^{-5.4}$



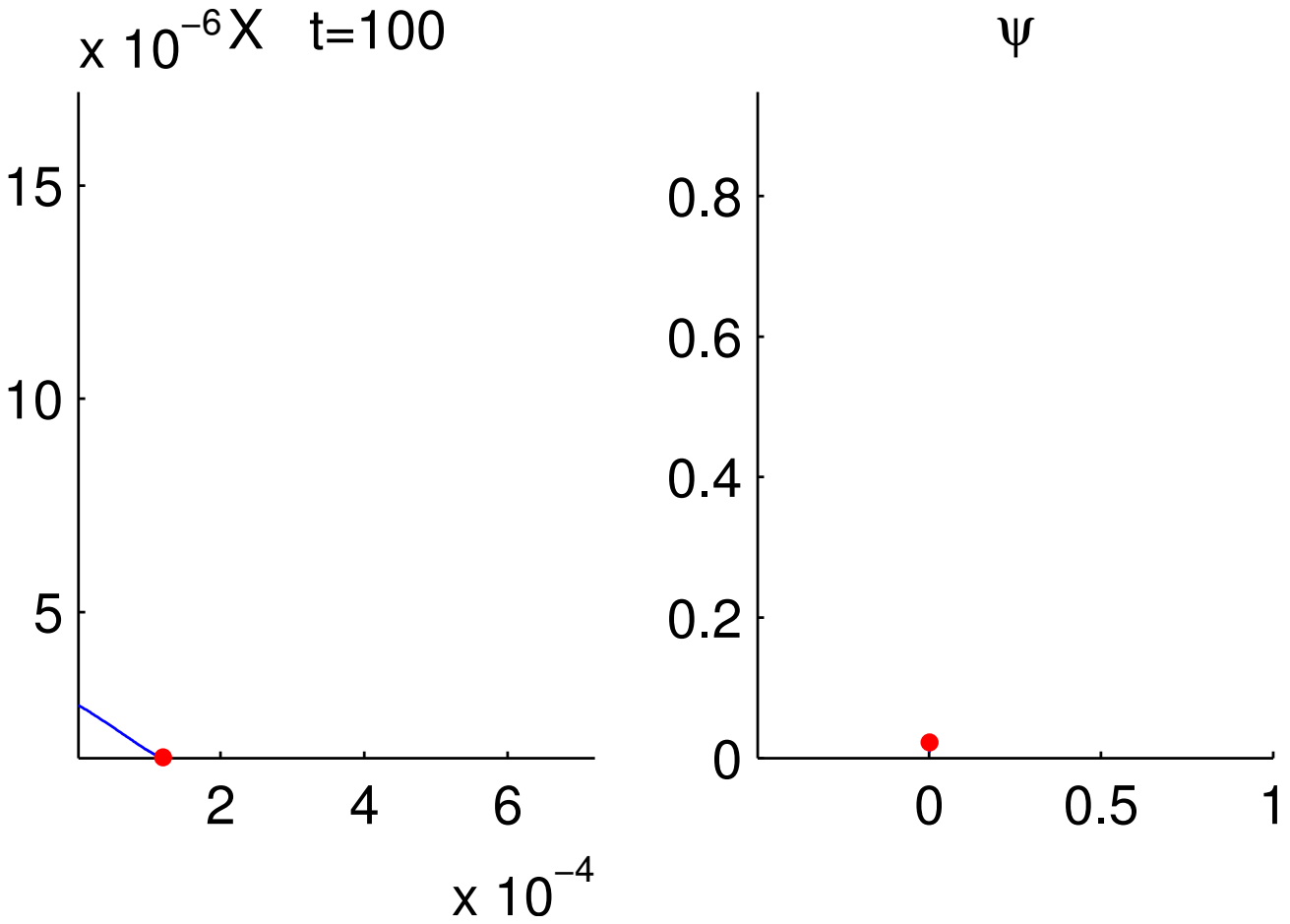
Transition to turbulence $\|X_0\| = 10^{-5.4}$



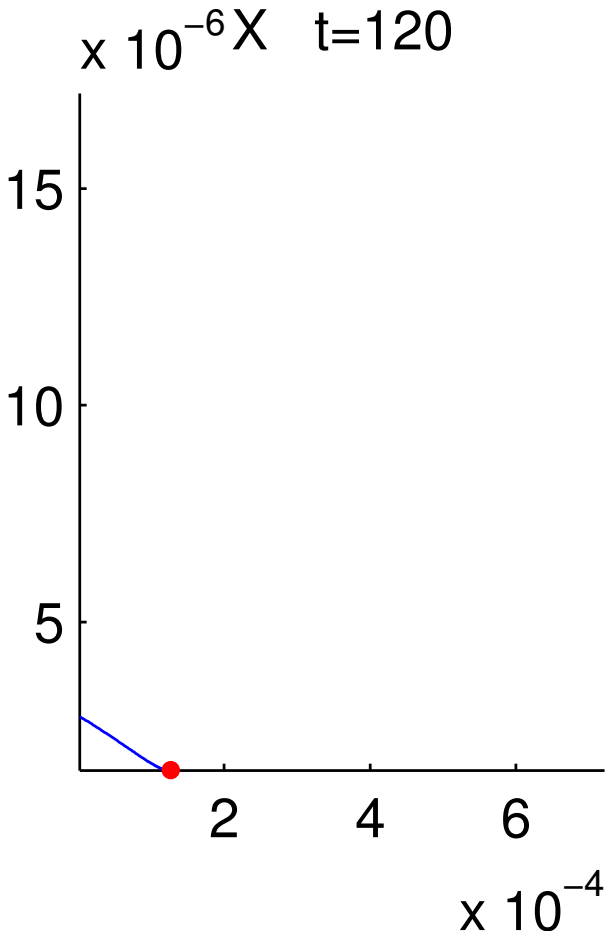
Transition to turbulence $\|X_0\| = 10^{-5.4}$

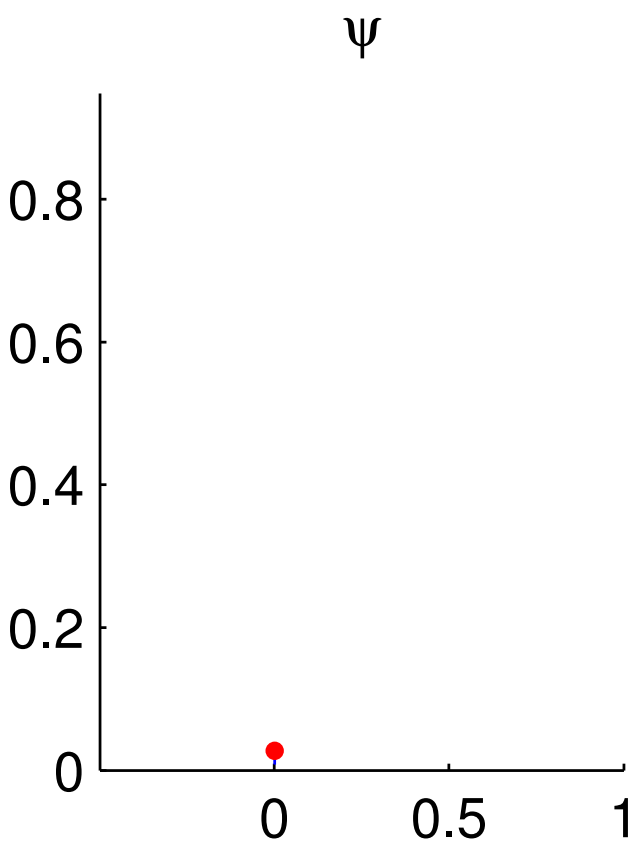


Transition to turbulence $\|X_0\| = 10^{-5.4}$

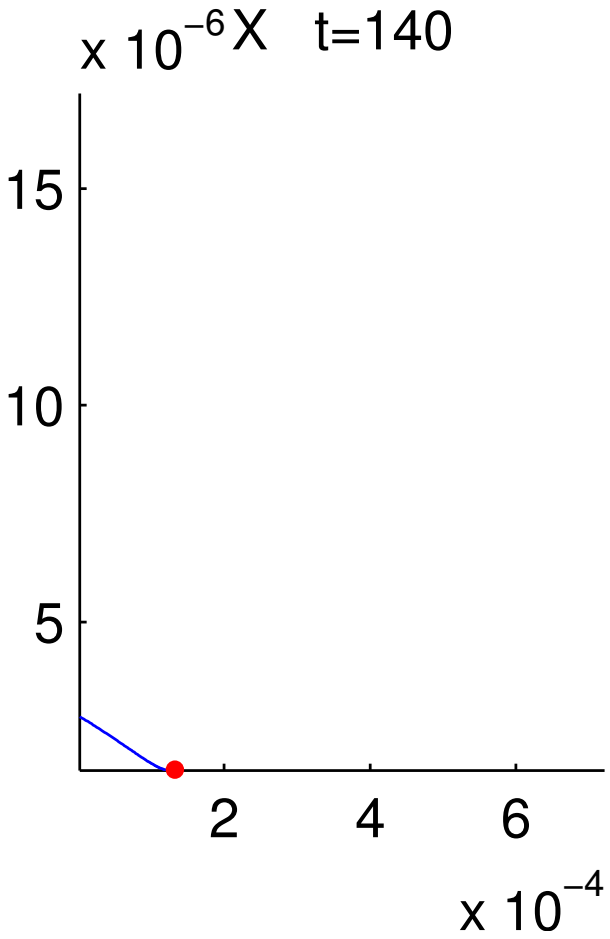


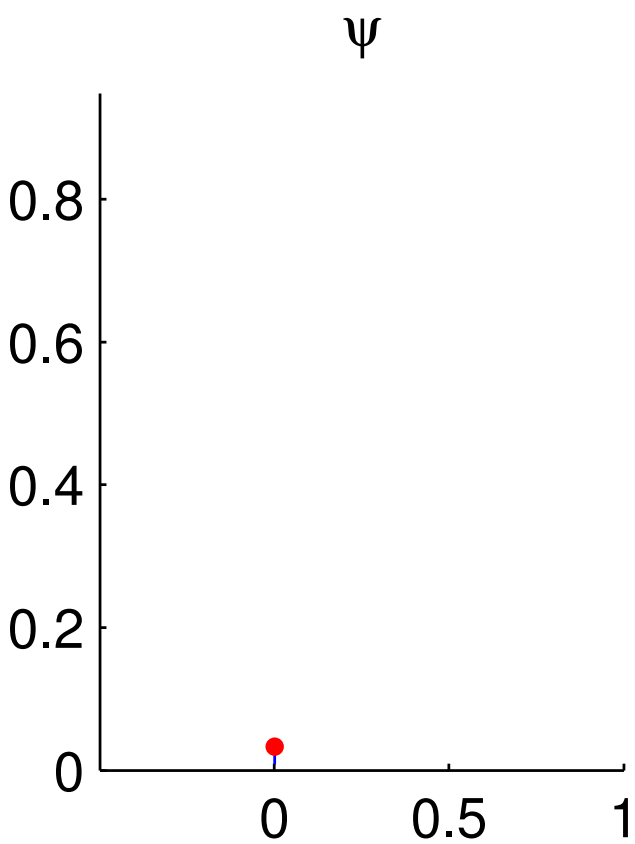
Transition to turbulence $\|X_0\| = 10^{-5.4}$



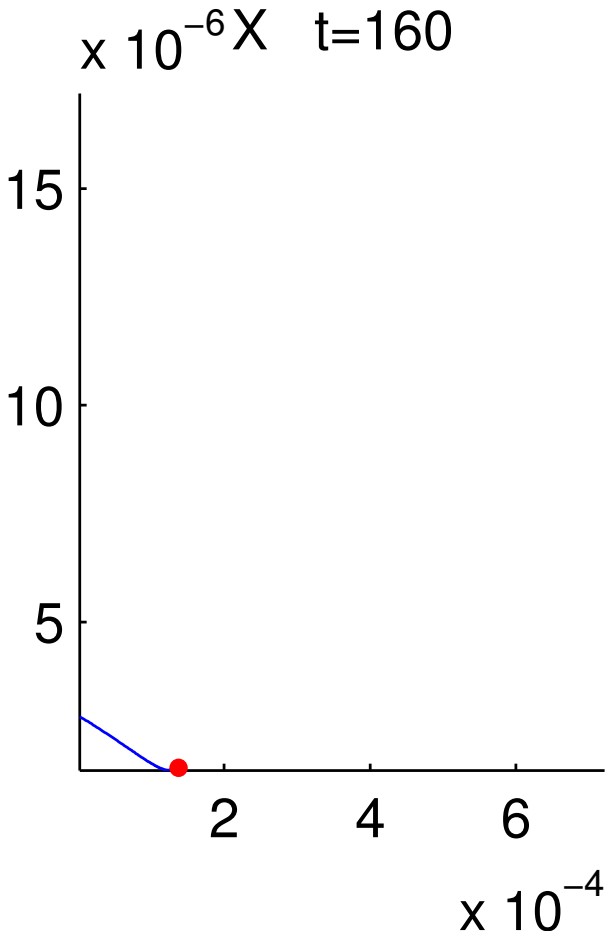


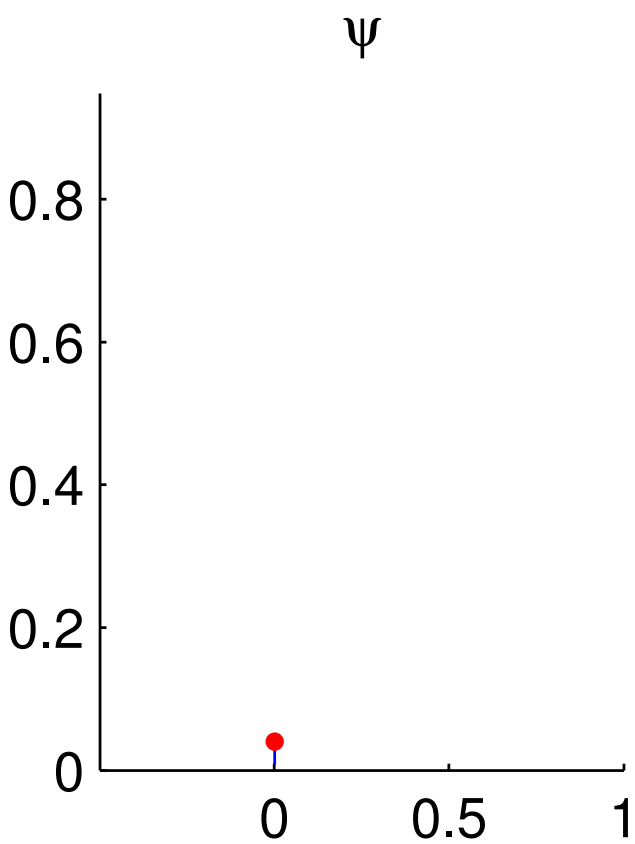
Transition to turbulence $\|X_0\| = 10^{-5.4}$



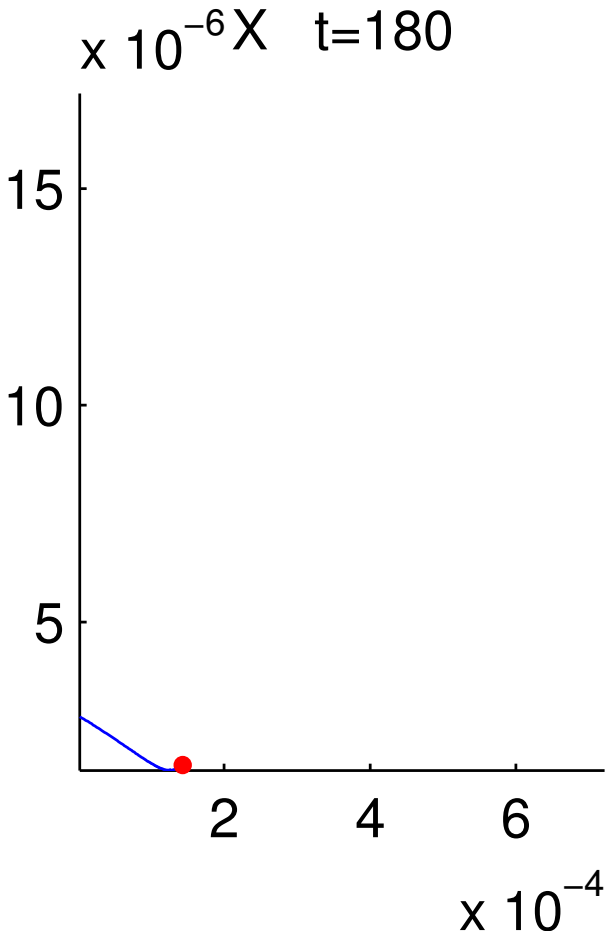


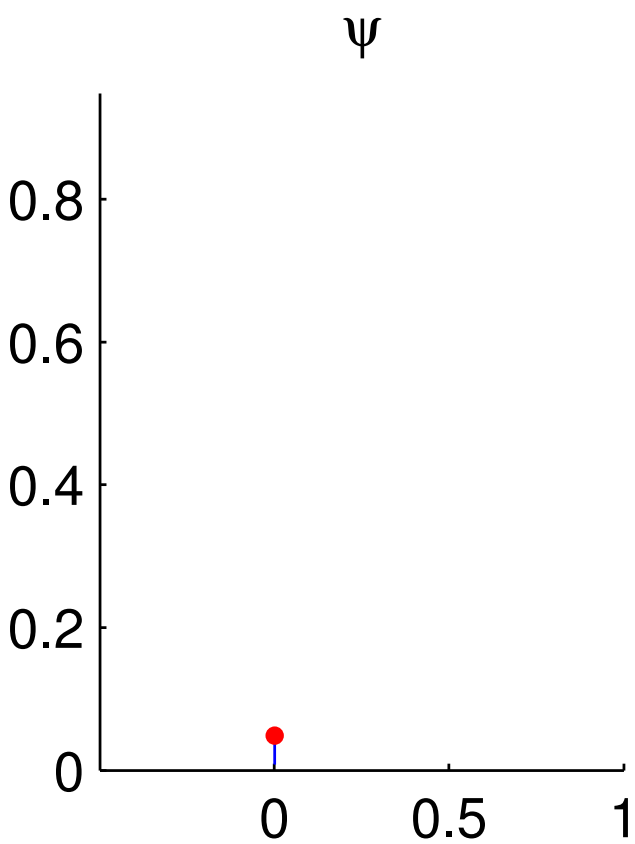
Transition to turbulence $\|X_0\| = 10^{-5.4}$



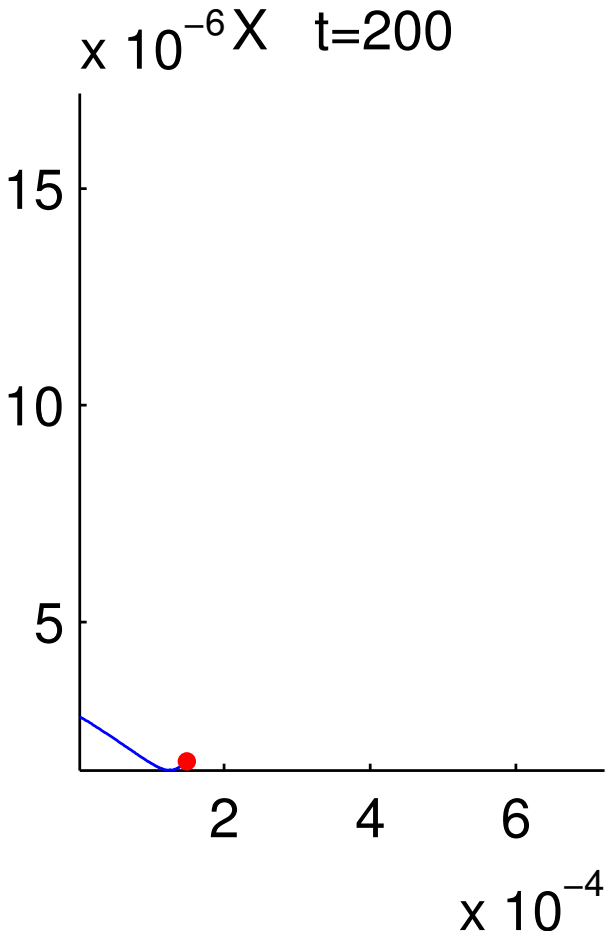


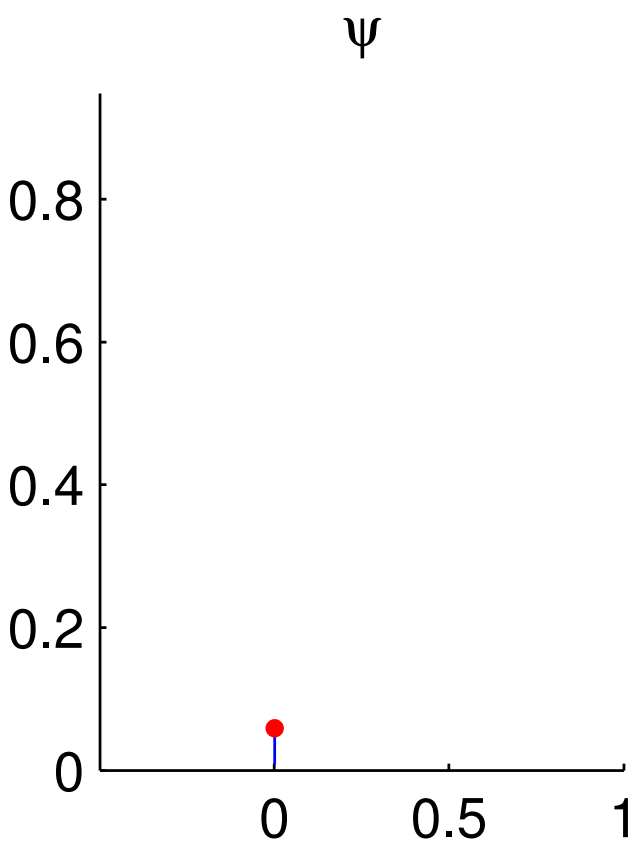
Transition to turbulence $\|X_0\| = 10^{-5.4}$



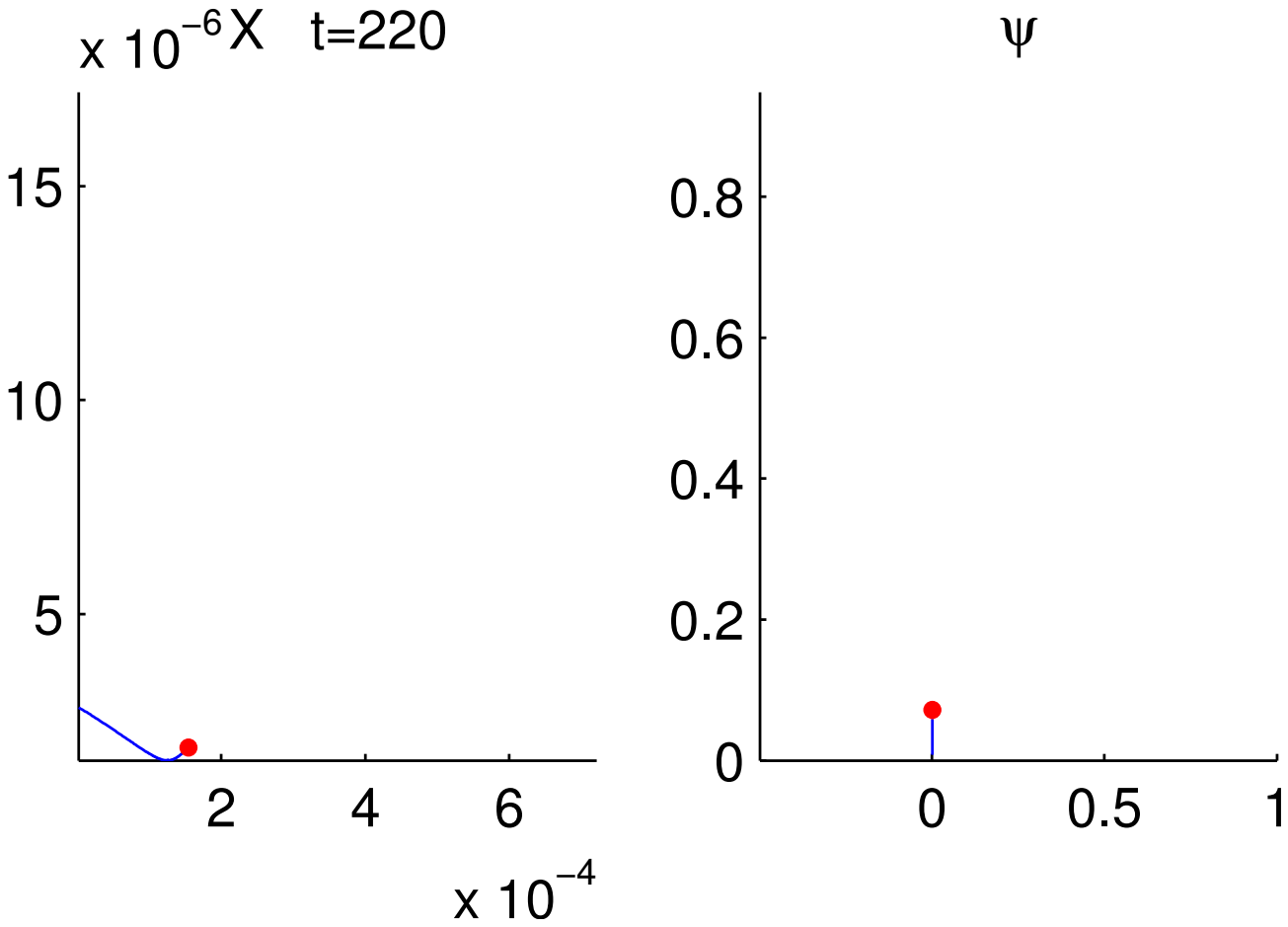


Transition to turbulence $\|X_0\| = 10^{-5.4}$

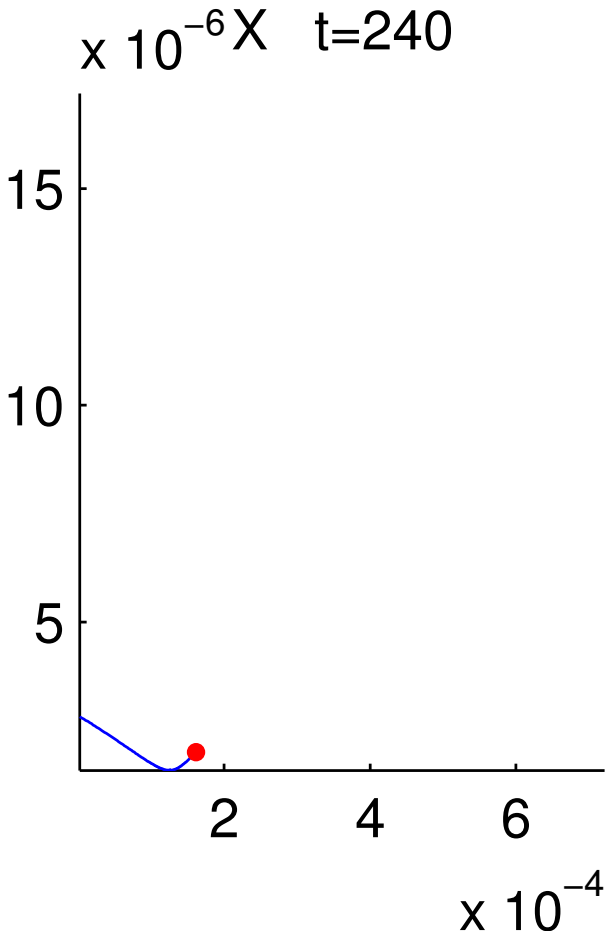


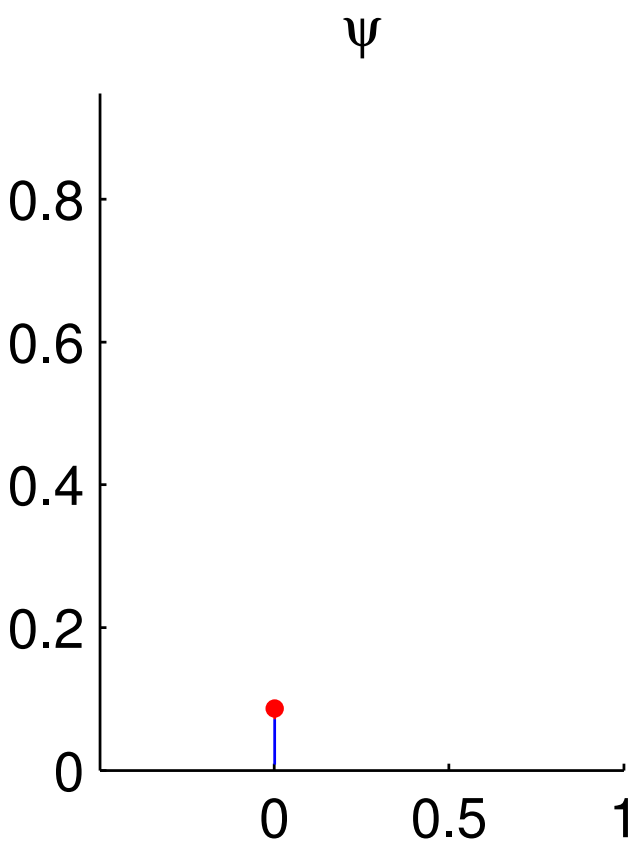


Transition to turbulence $\|X_0\| = 10^{-5.4}$

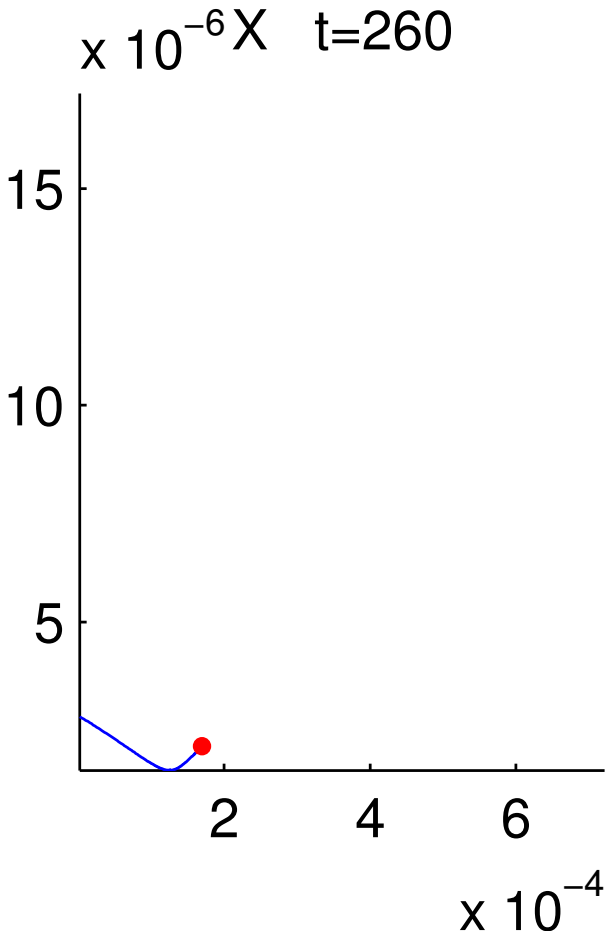


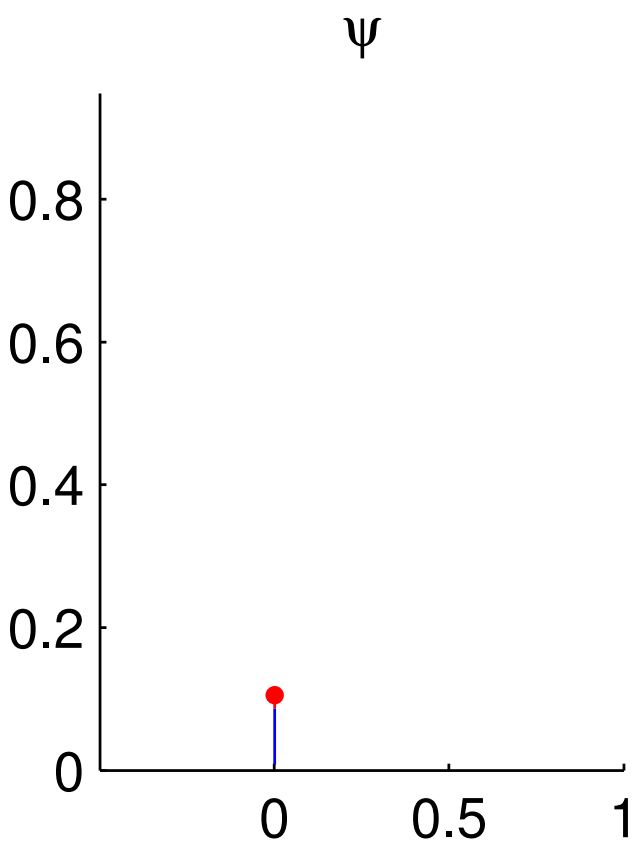
Transition to turbulence $\|X_0\| = 10^{-5.4}$



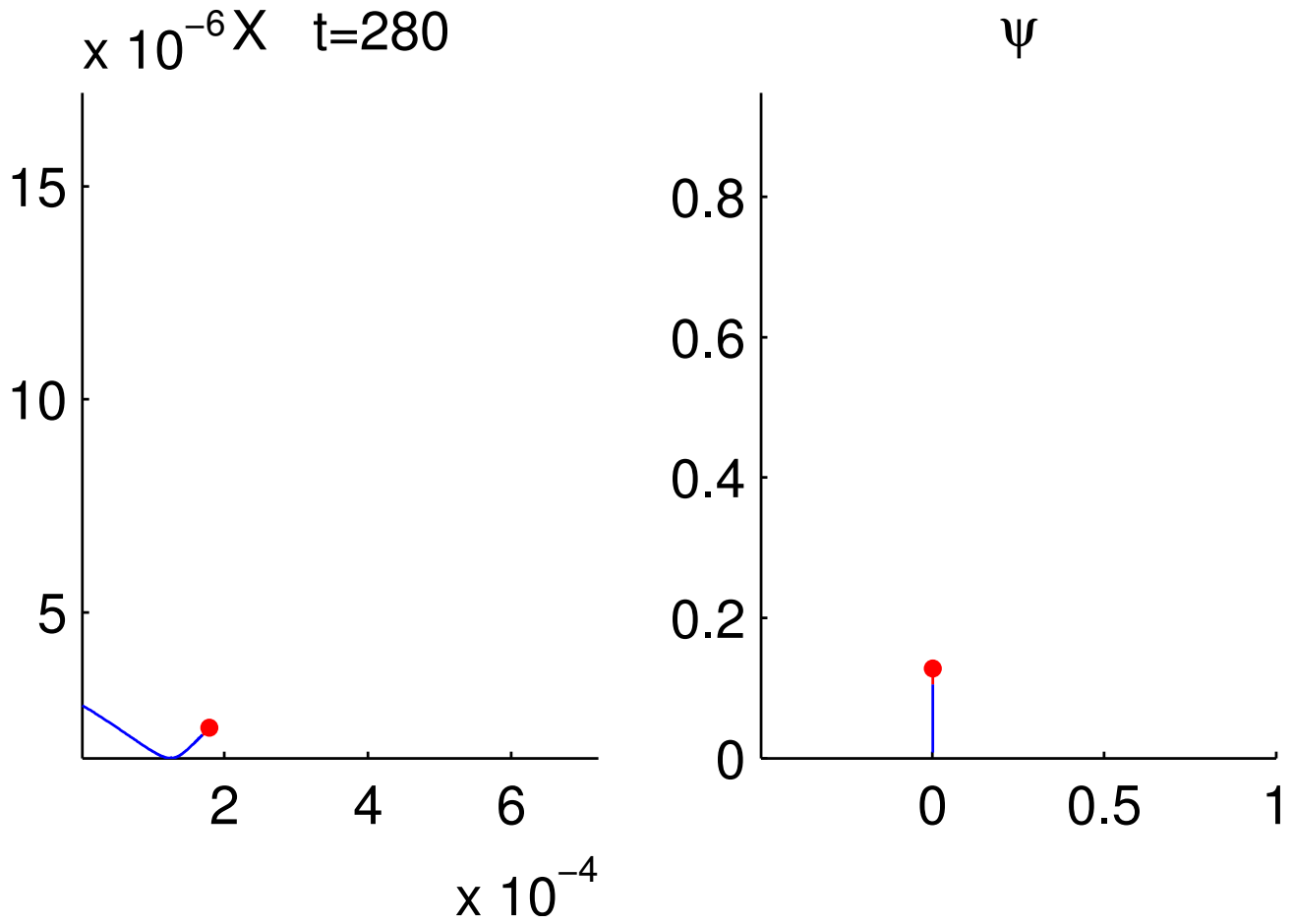


Transition to turbulence $\|X_0\| = 10^{-5.4}$

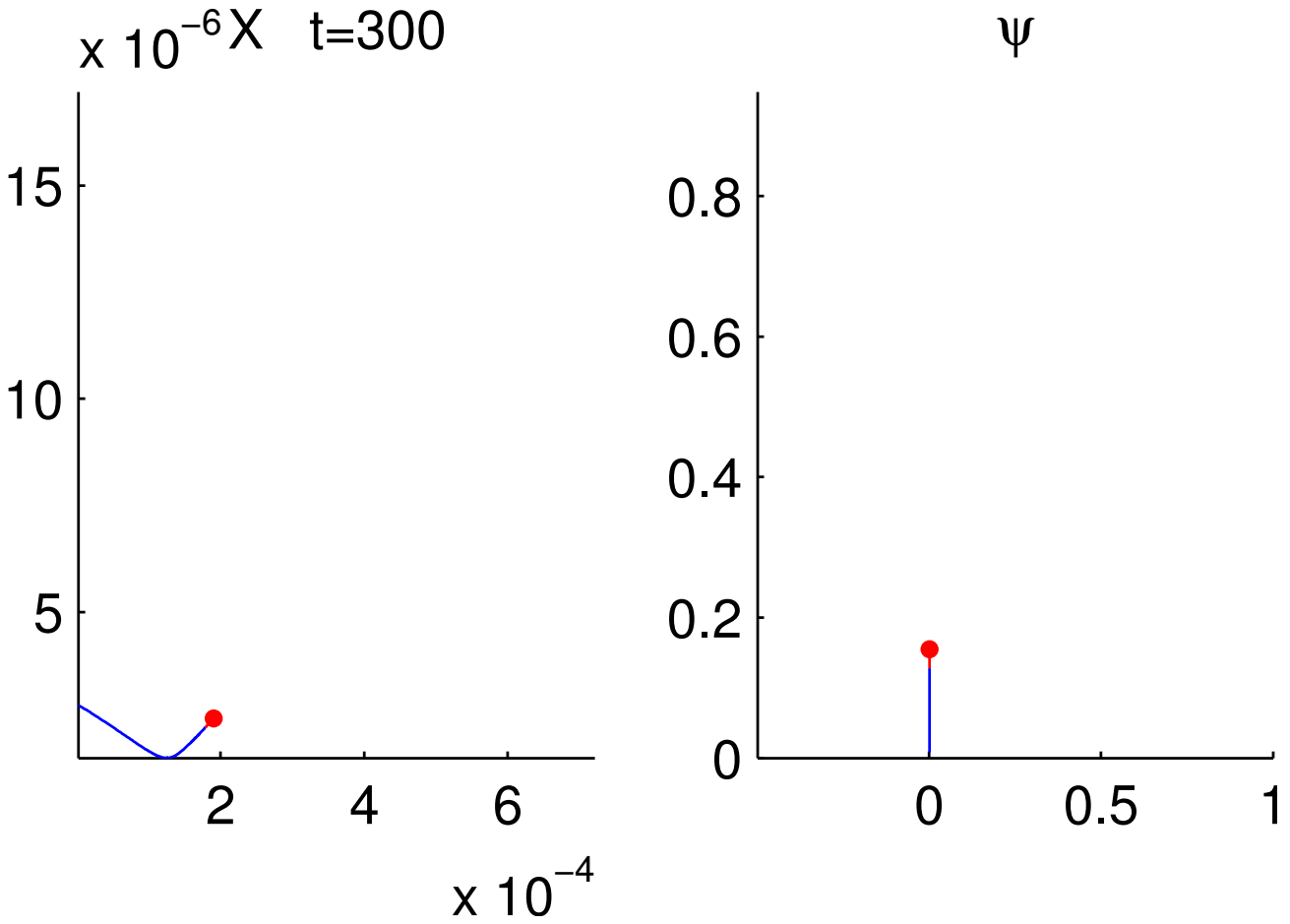




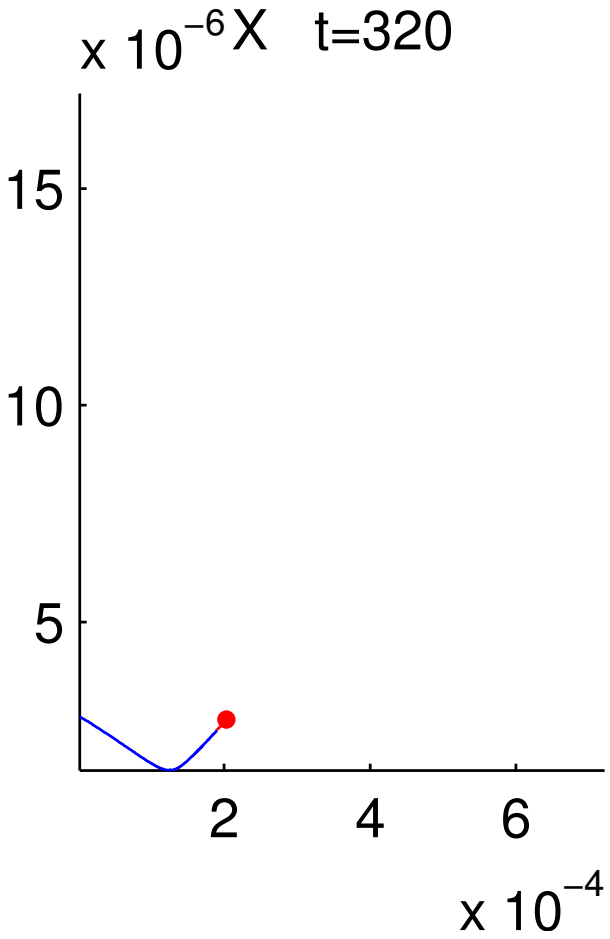
Transition to turbulence $\|X_0\| = 10^{-5.4}$

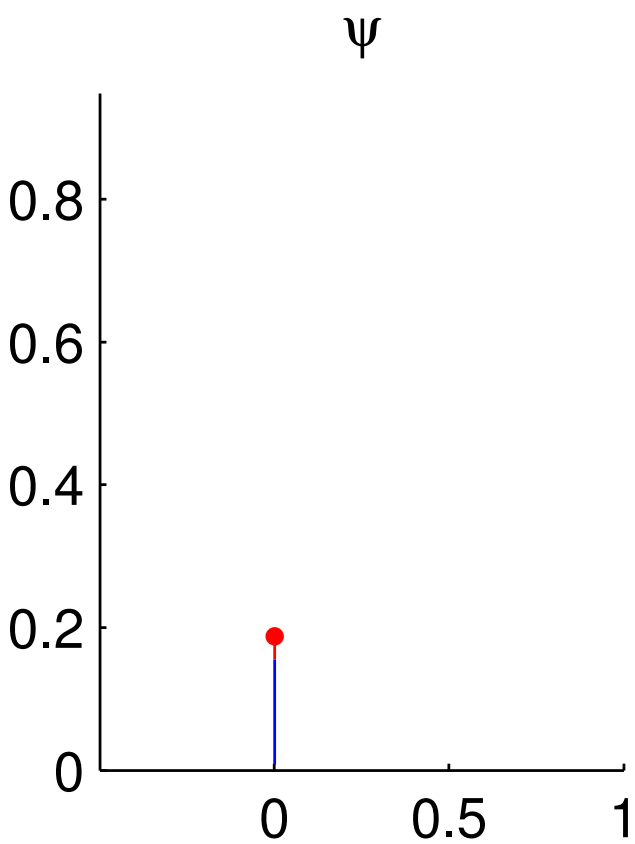


Transition to turbulence $\|X_0\| = 10^{-5.4}$

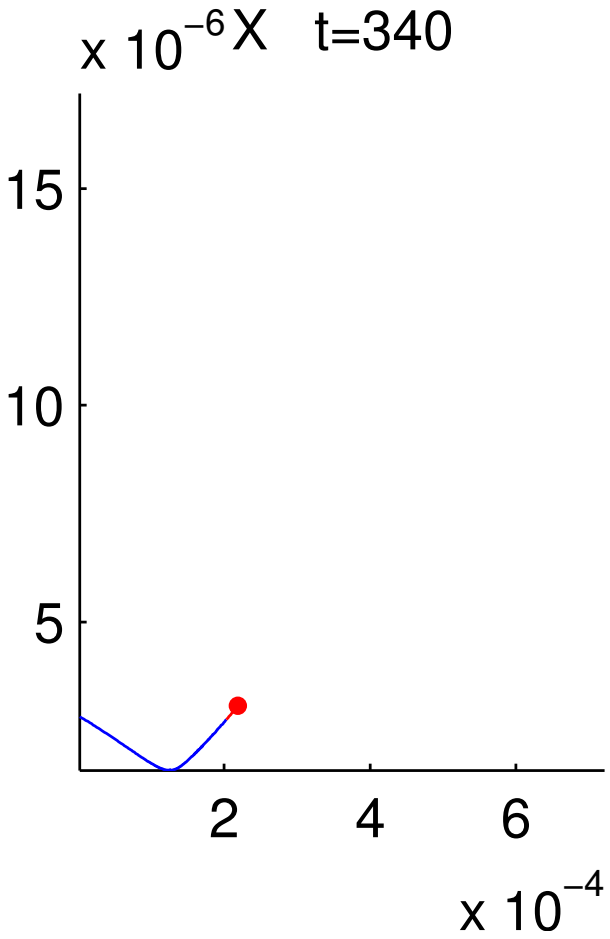


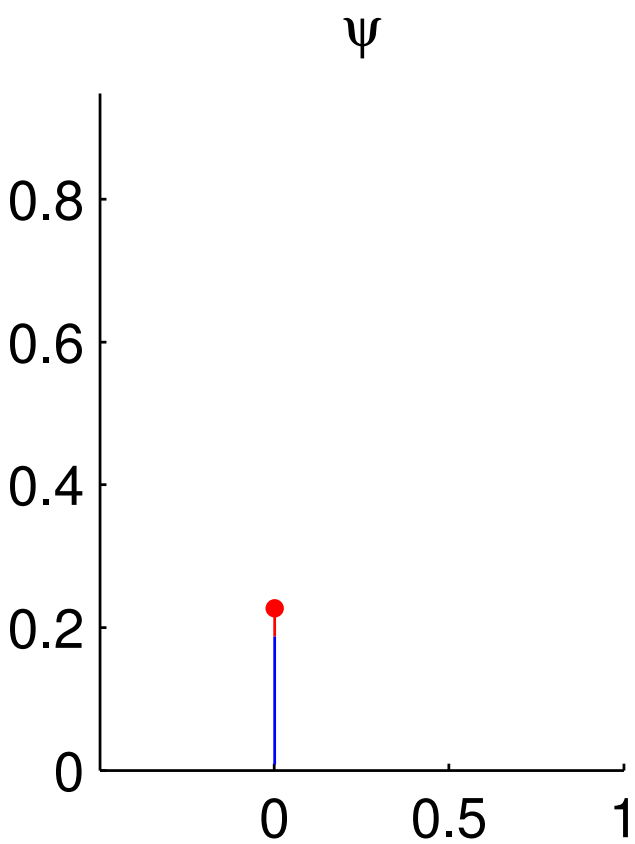
Transition to turbulence $\|X_0\| = 10^{-5.4}$



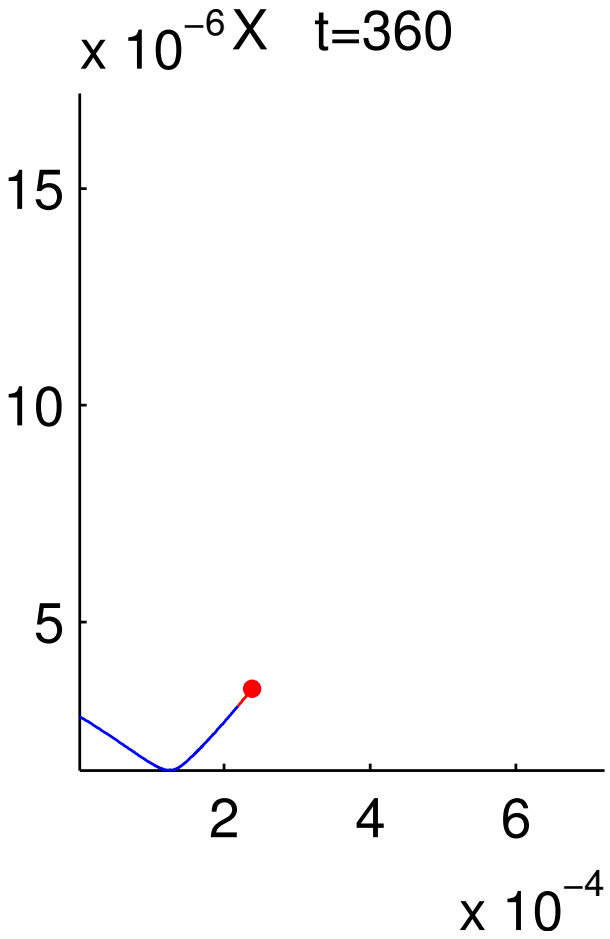


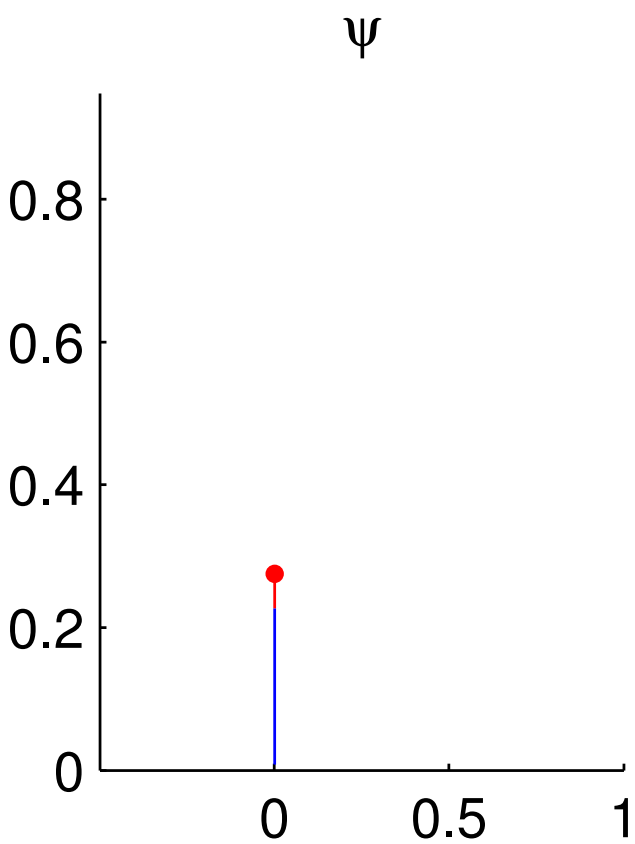
Transition to turbulence $\|X_0\| = 10^{-5.4}$



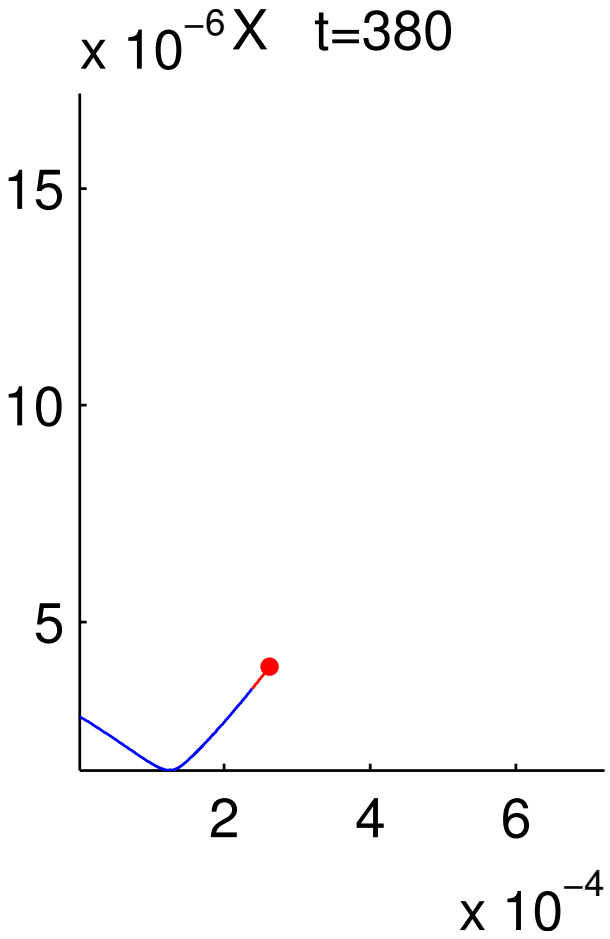


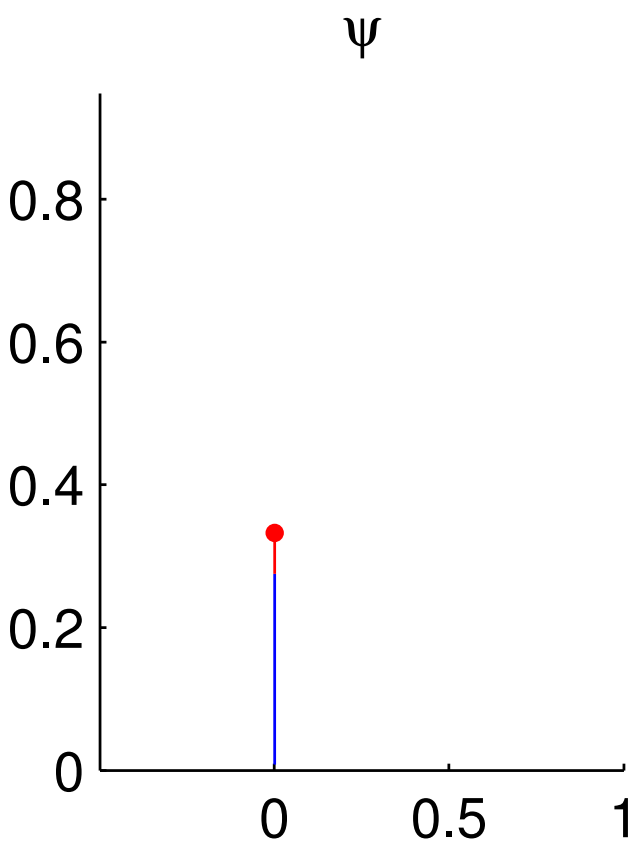
Transition to turbulence $\|X_0\| = 10^{-5.4}$



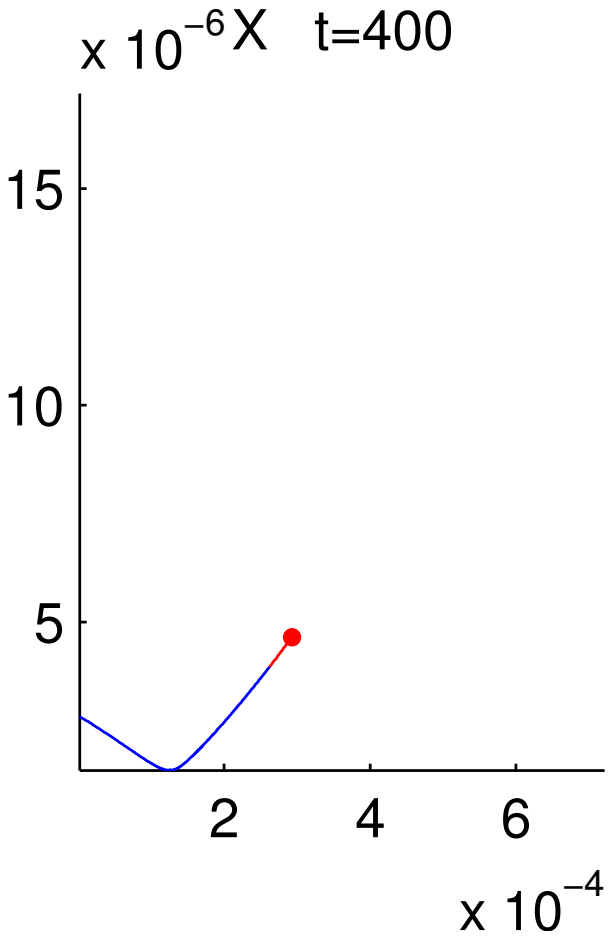


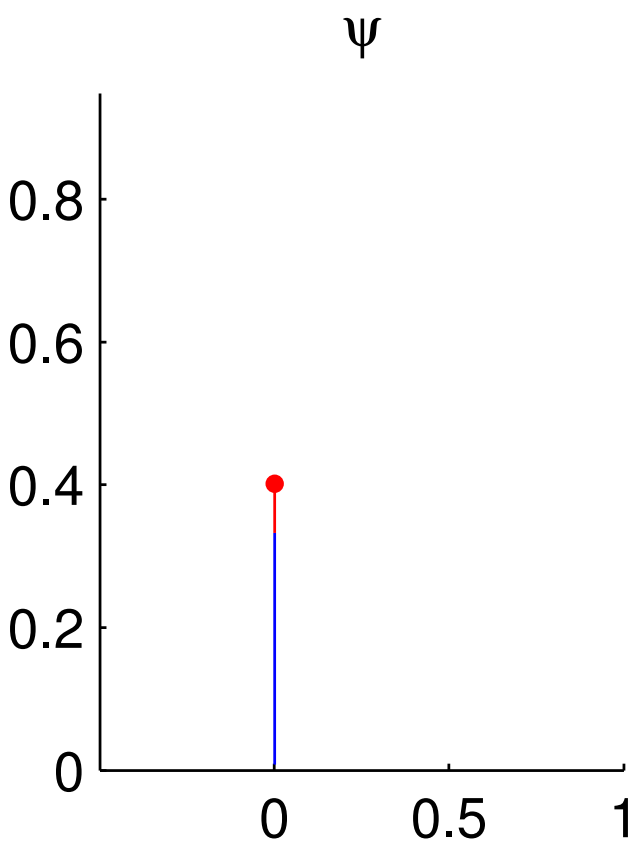
Transition to turbulence $\|X_0\| = 10^{-5.4}$



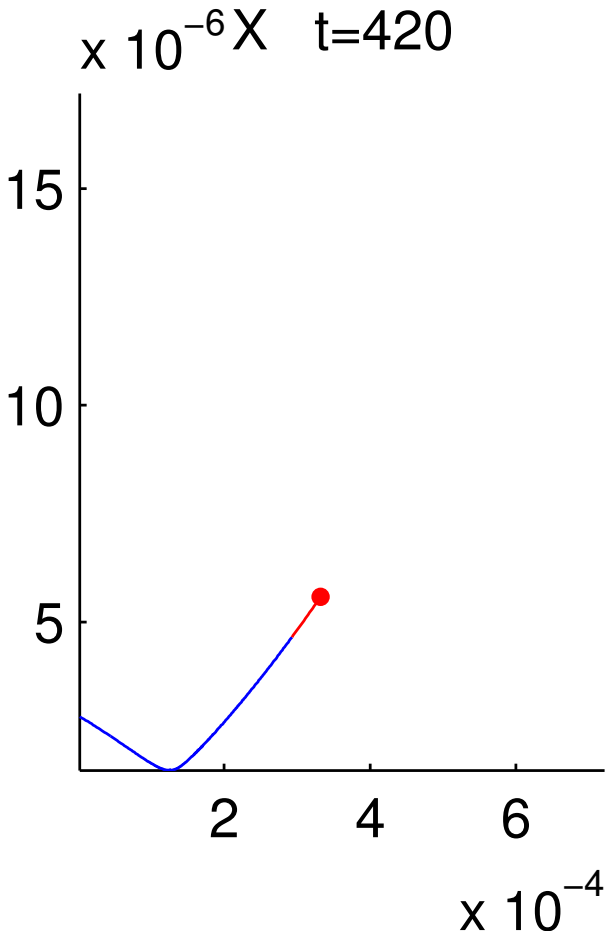


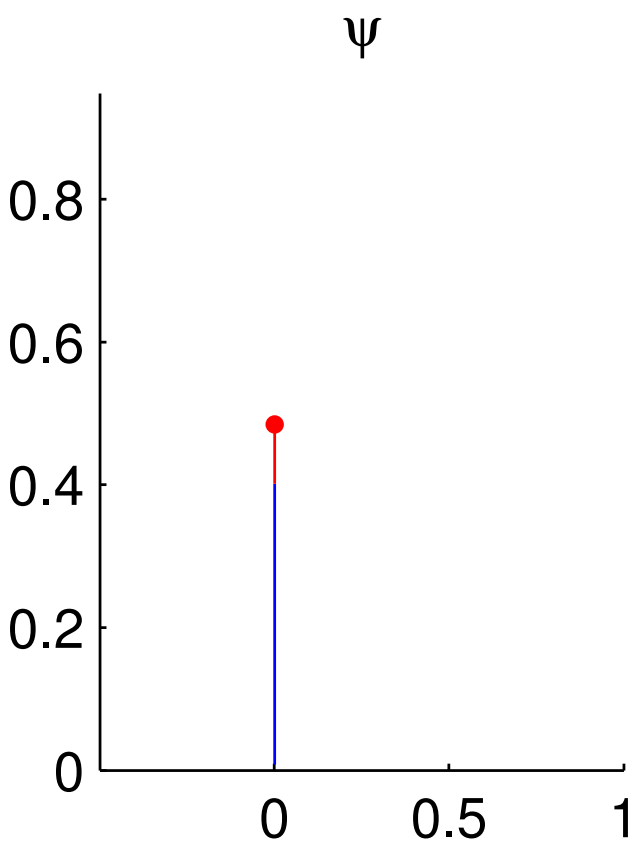
Transition to turbulence $\|X_0\| = 10^{-5.4}$



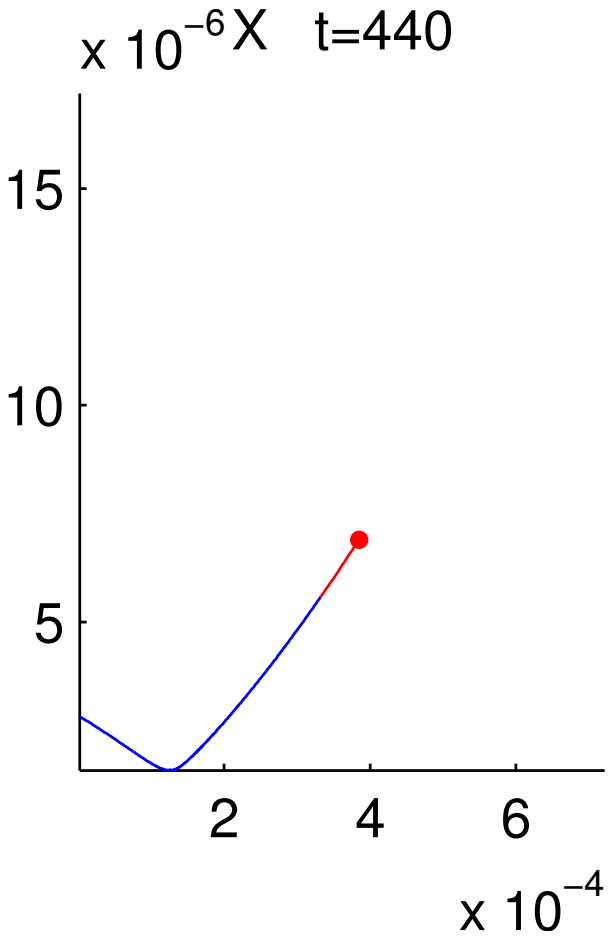


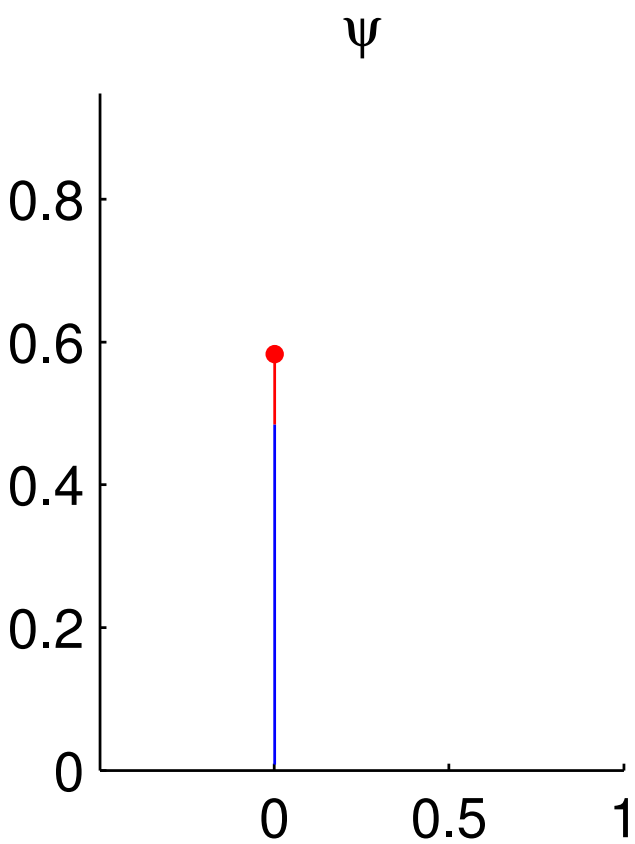
Transition to turbulence $\|X_0\| = 10^{-5.4}$



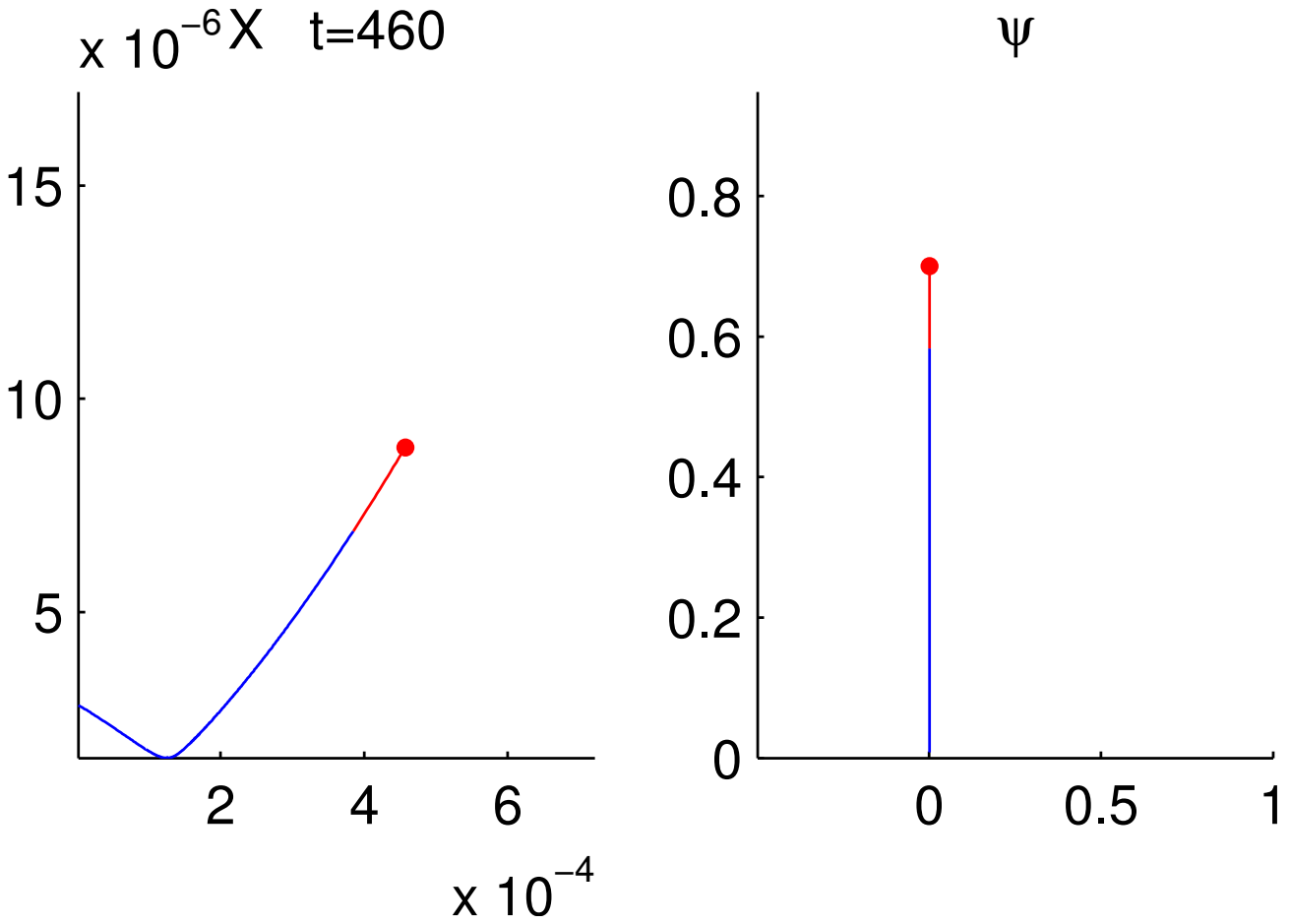


Transition to turbulence $\|X_0\| = 10^{-5.4}$

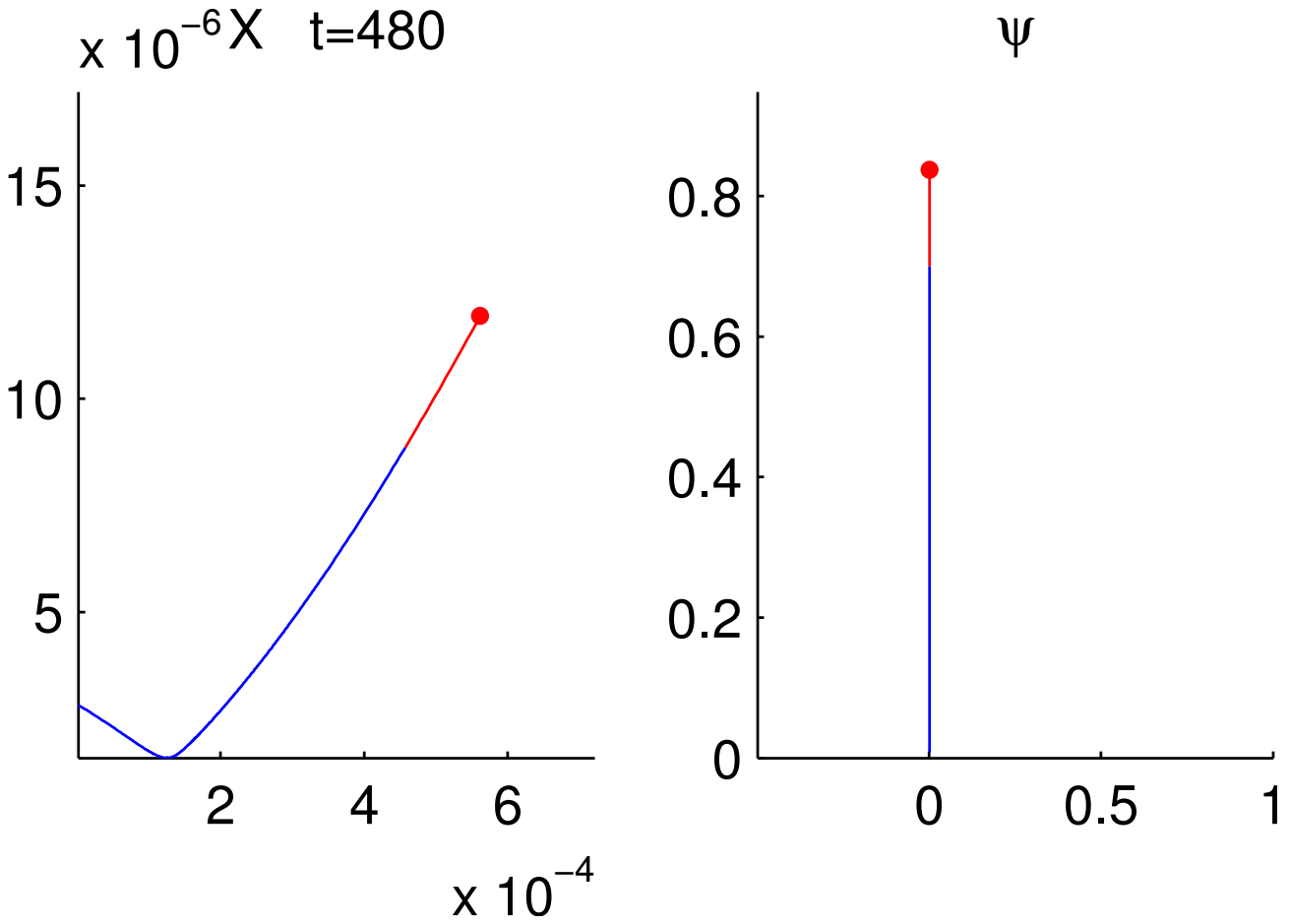




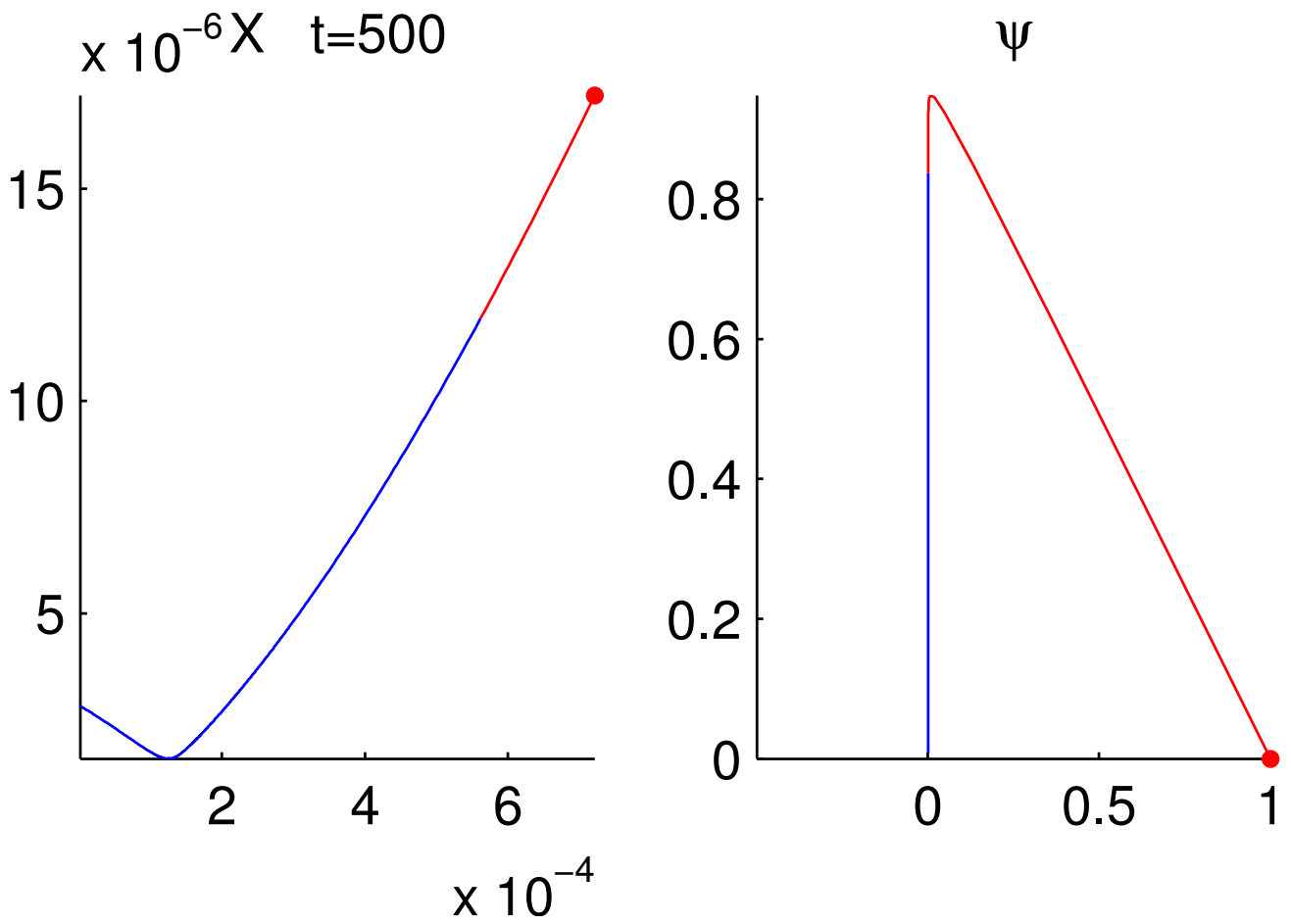
Transition to turbulence $\|X_0\| = 10^{-5.4}$



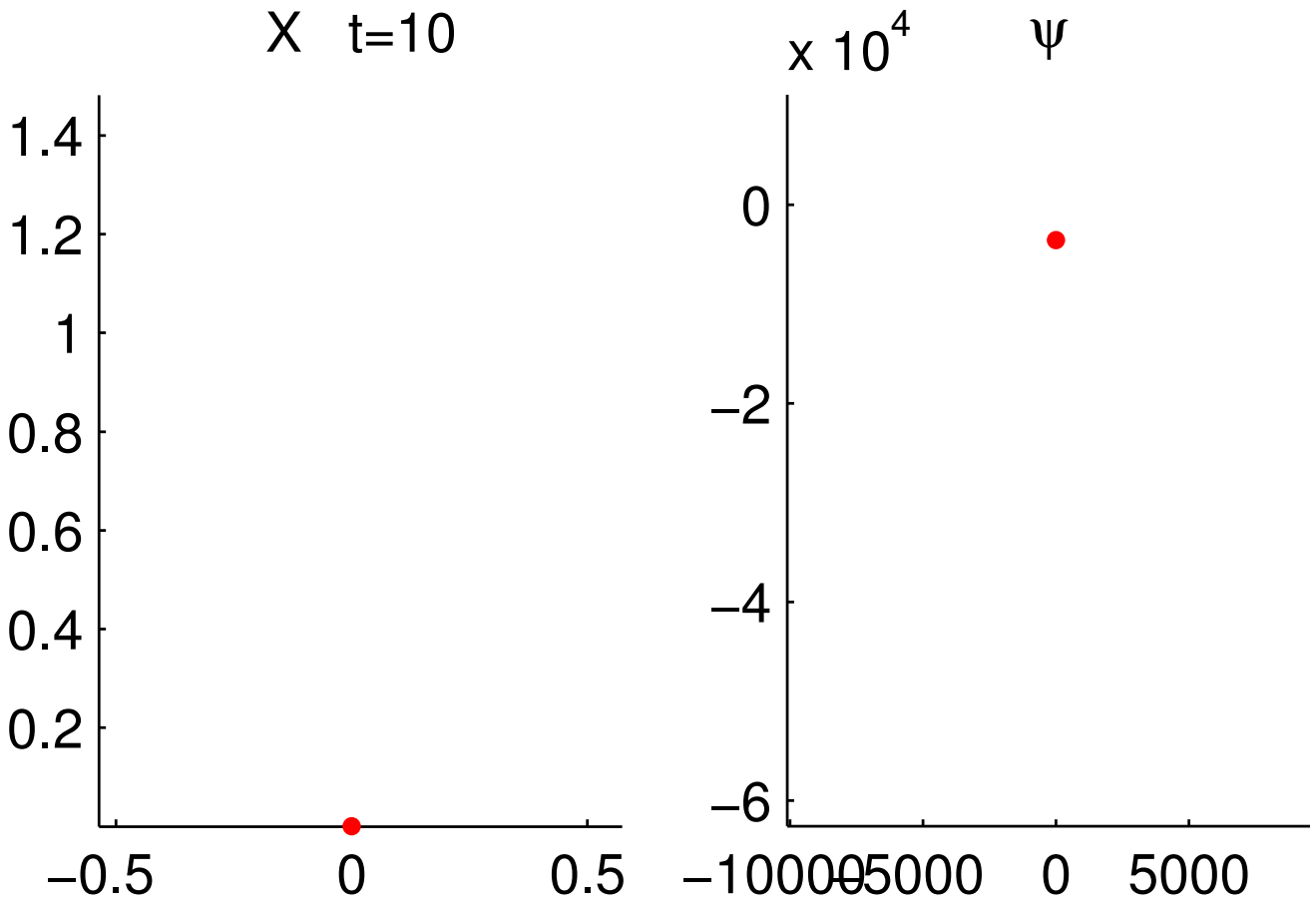
Transition to turbulence $\|X_0\| = 10^{-5.4}$



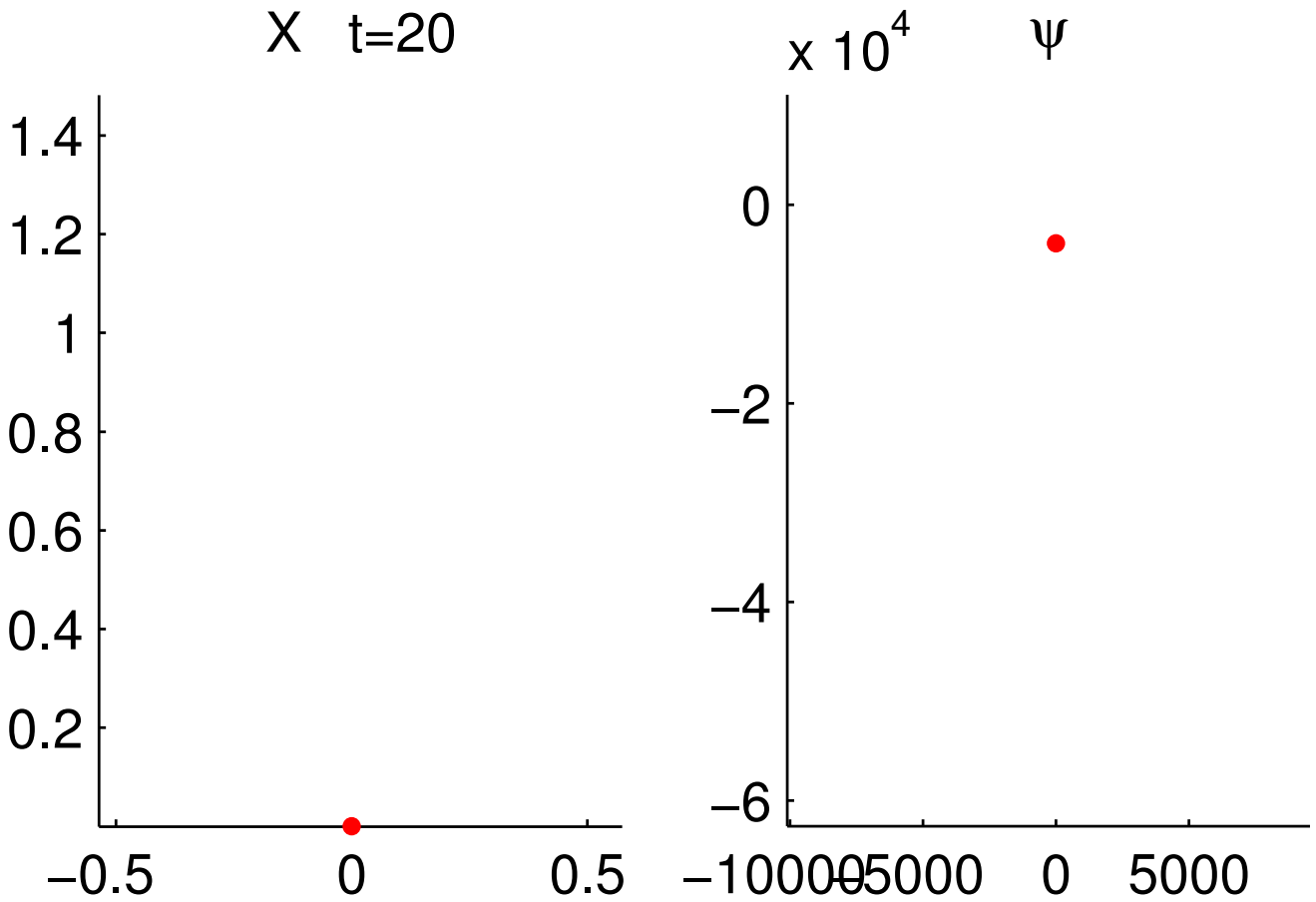
Transition to turbulence $\|X_0\| = 10^{-5.4}$



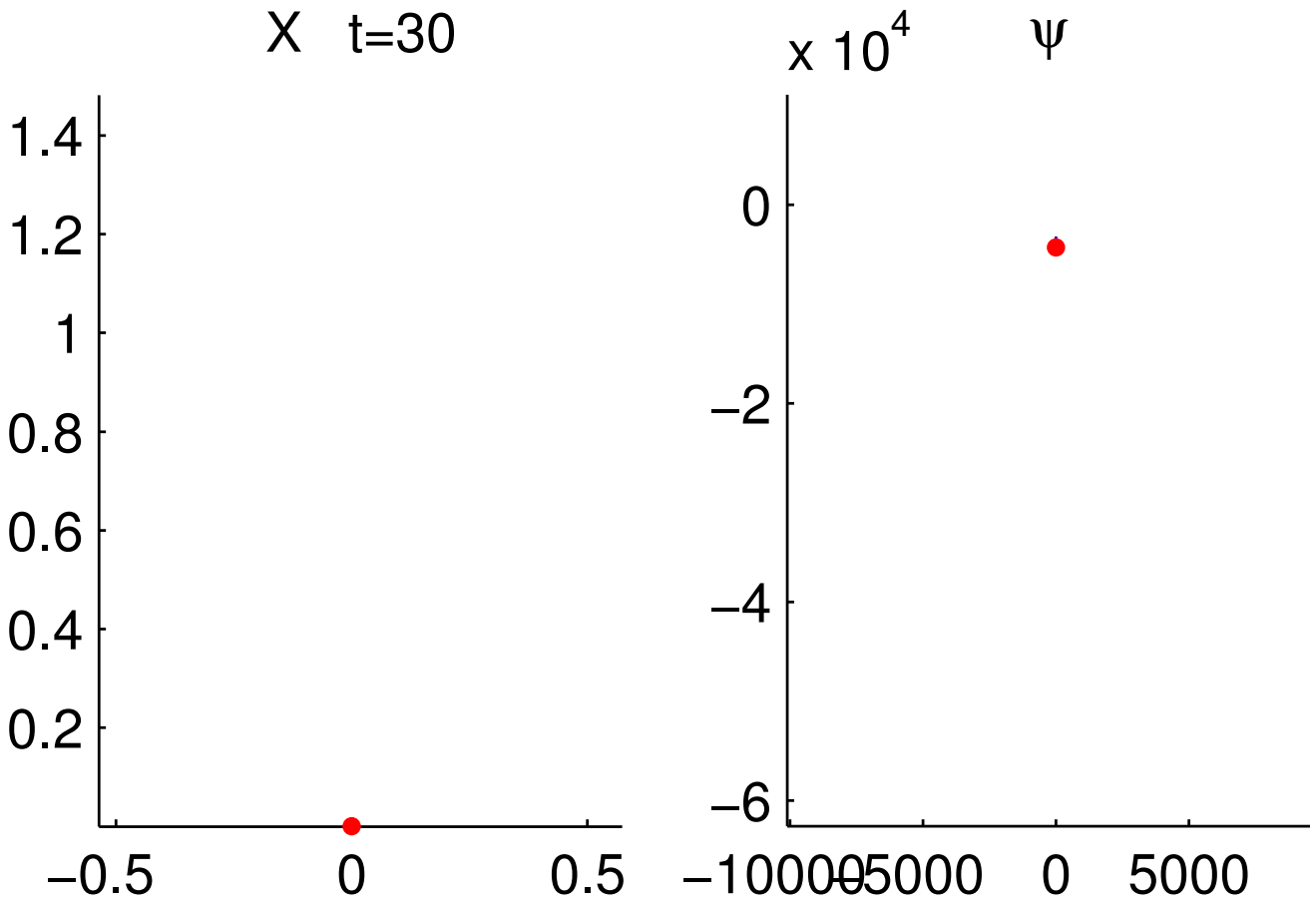
Transition to turbulence $\|X_0\| = 10^{-5.2}$



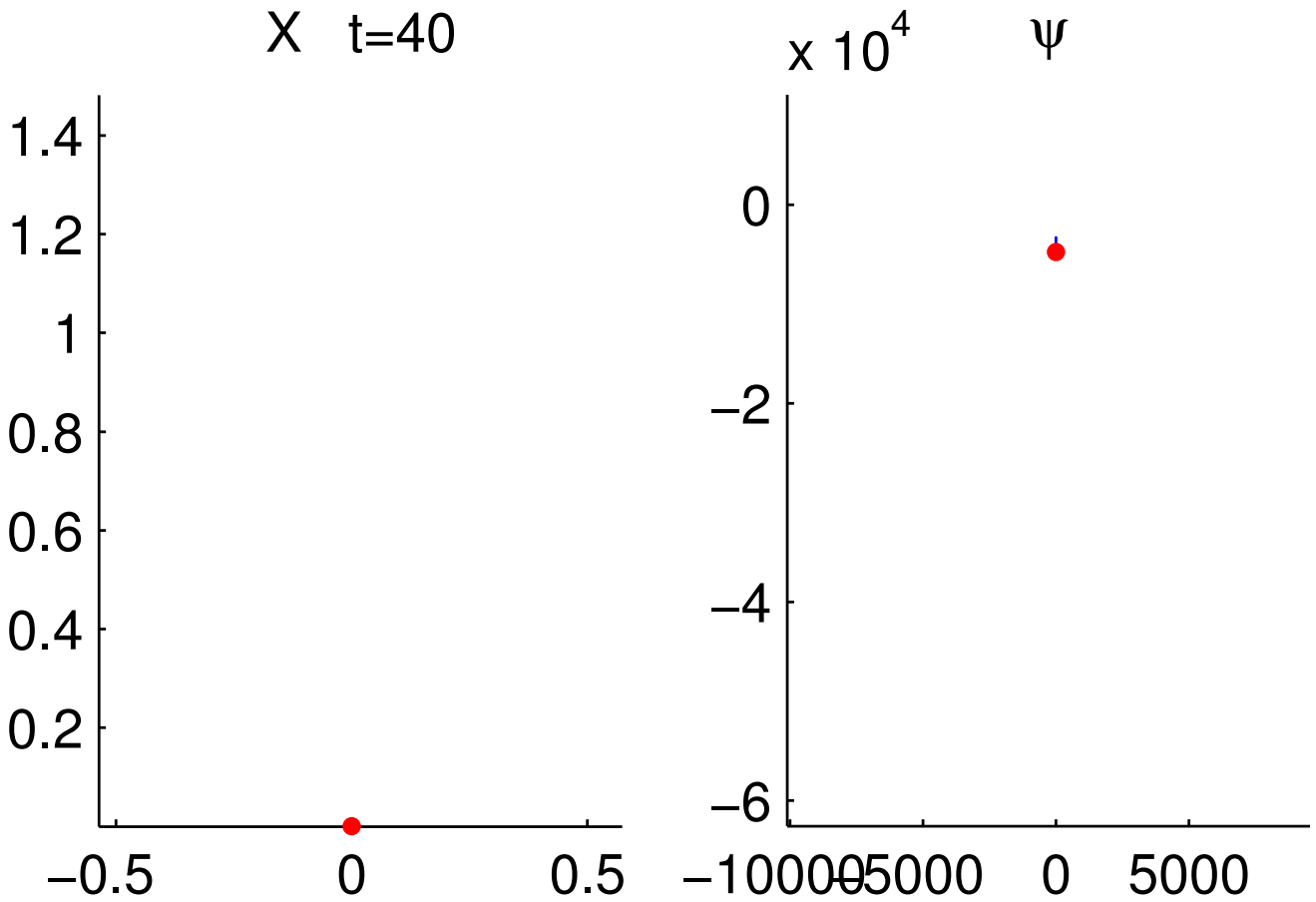
Transition to turbulence $\|X_0\| = 10^{-5.2}$



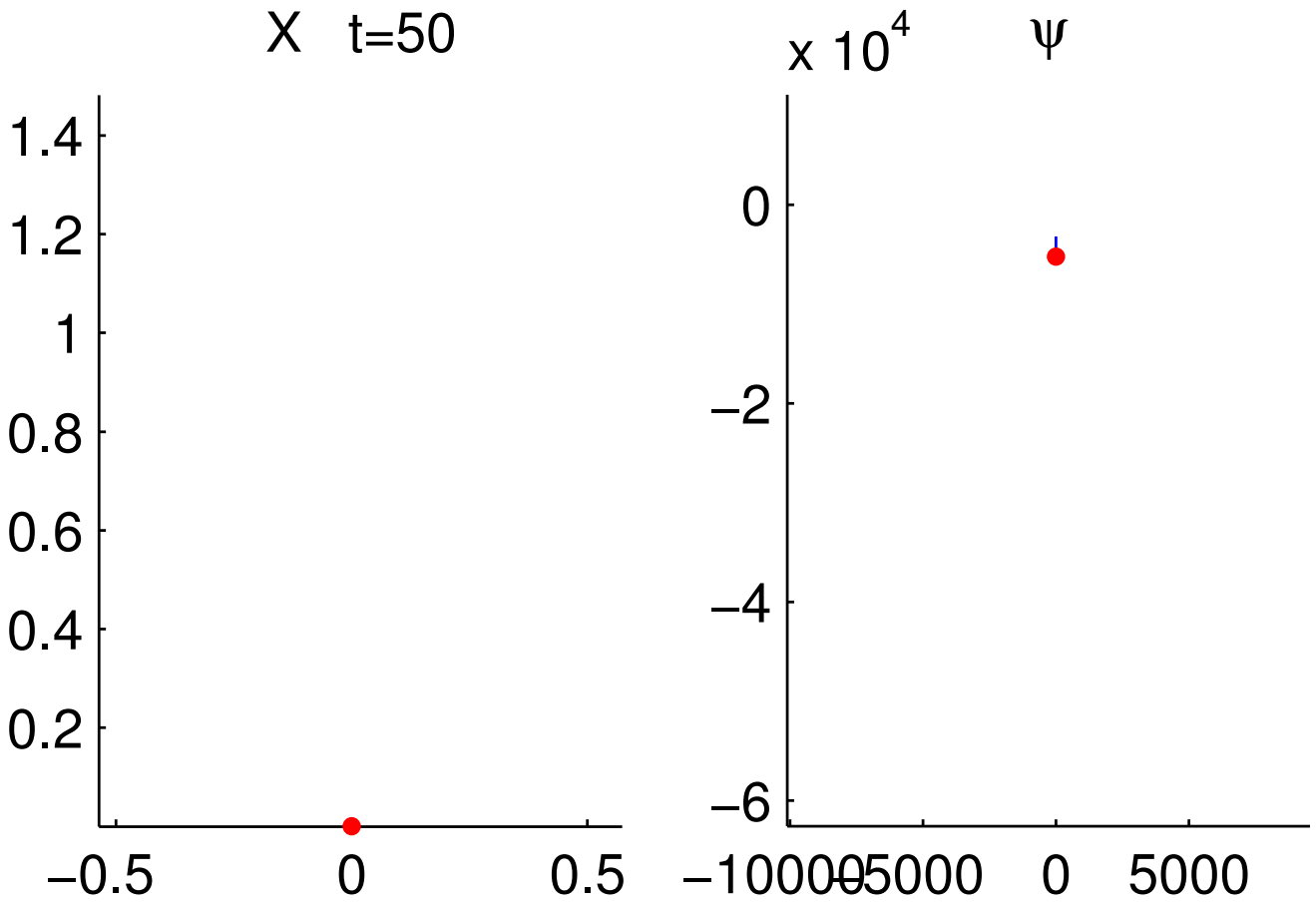
Transition to turbulence $\|X_0\| = 10^{-5.2}$



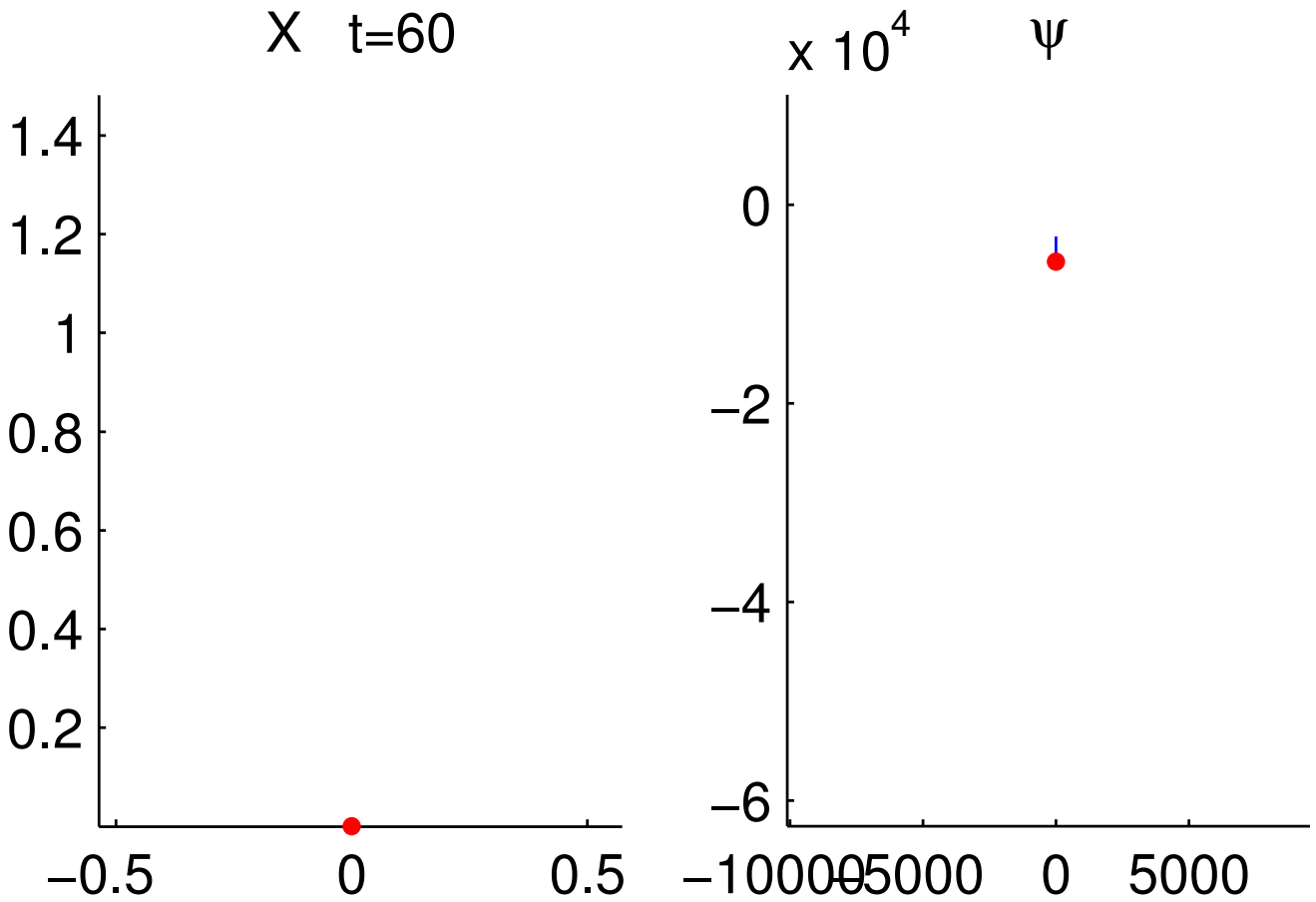
Transition to turbulence $\|X_0\| = 10^{-5.2}$



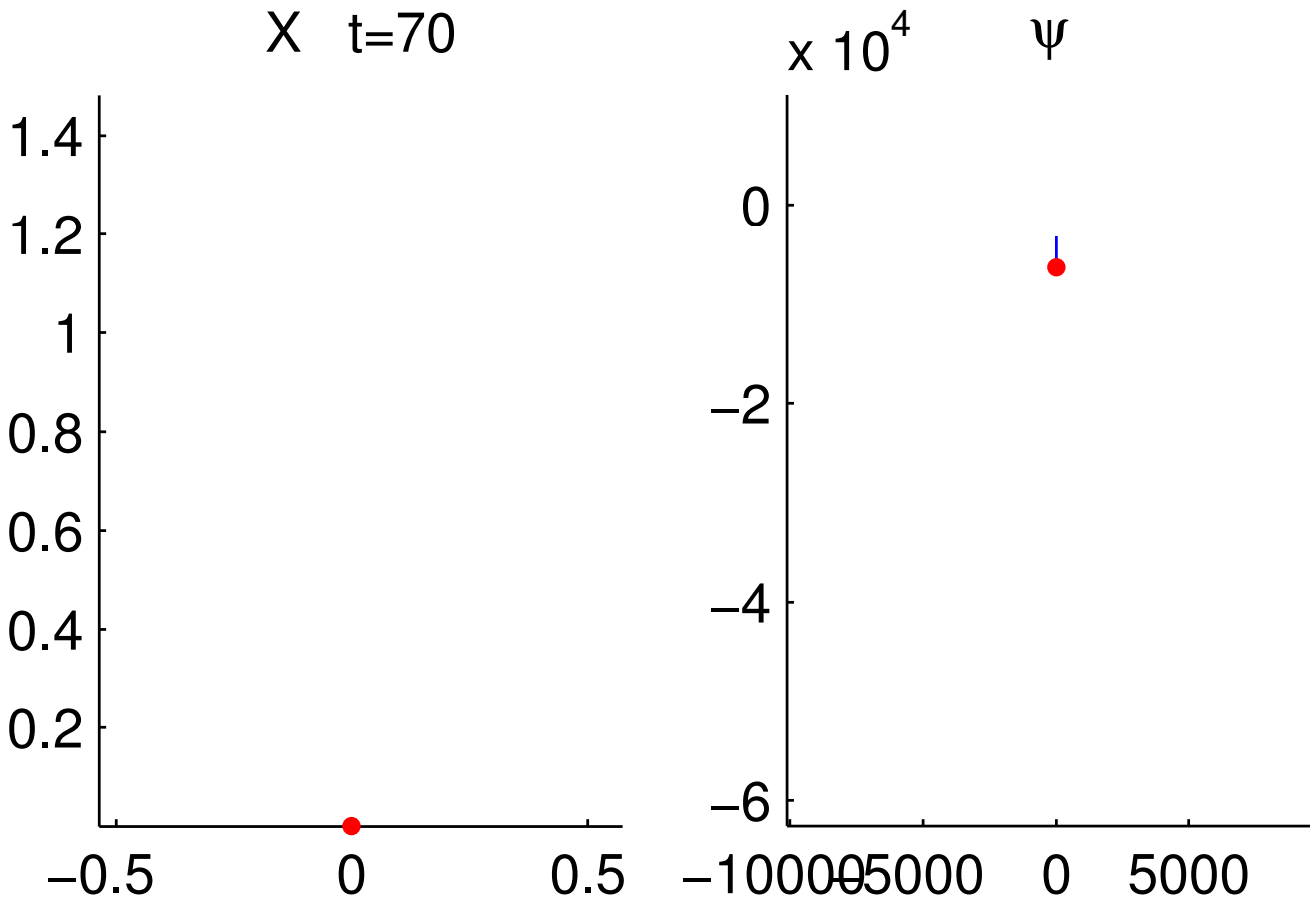
Transition to turbulence $\|X_0\| = 10^{-5.2}$



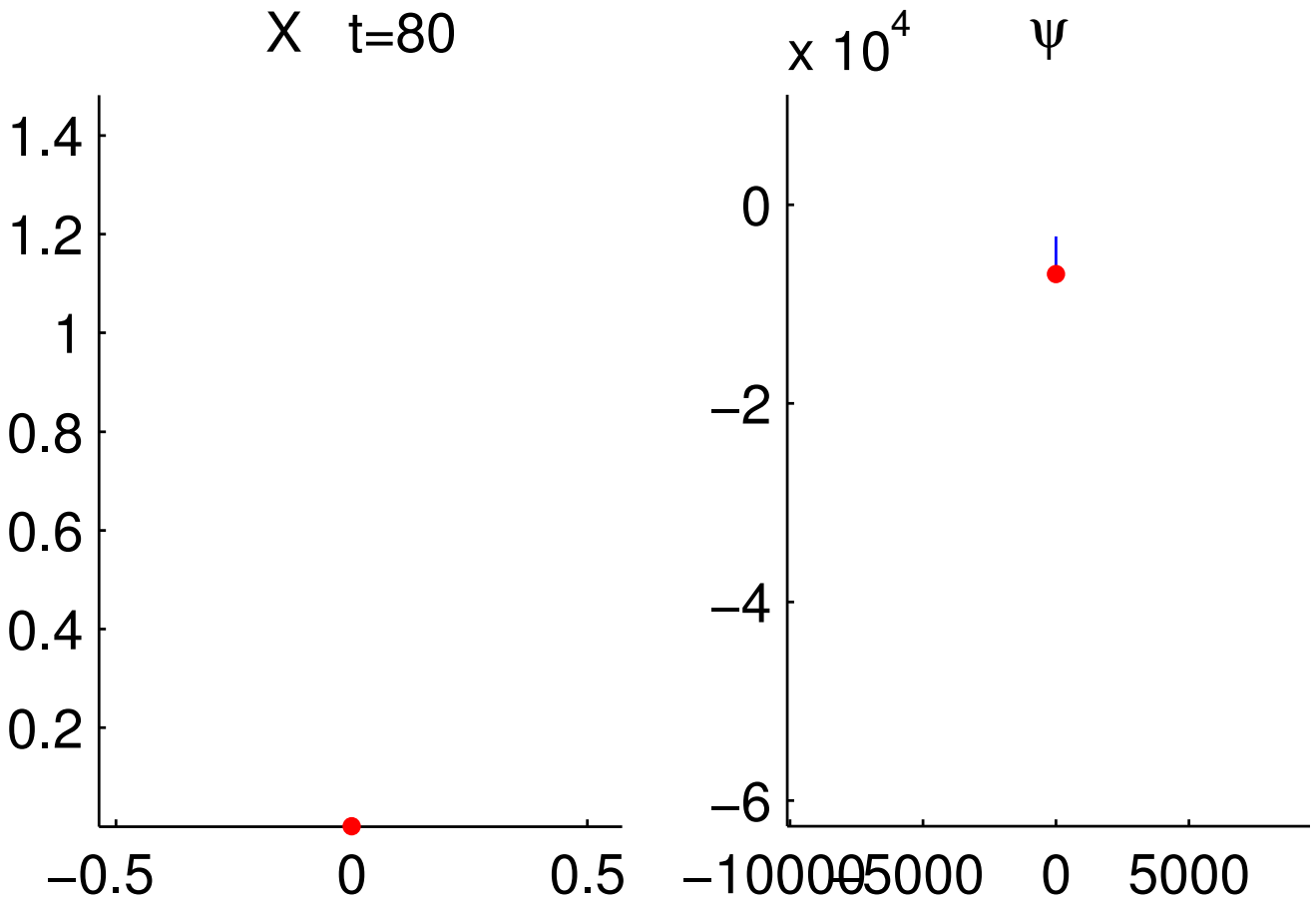
Transition to turbulence $\|X_0\| = 10^{-5.2}$



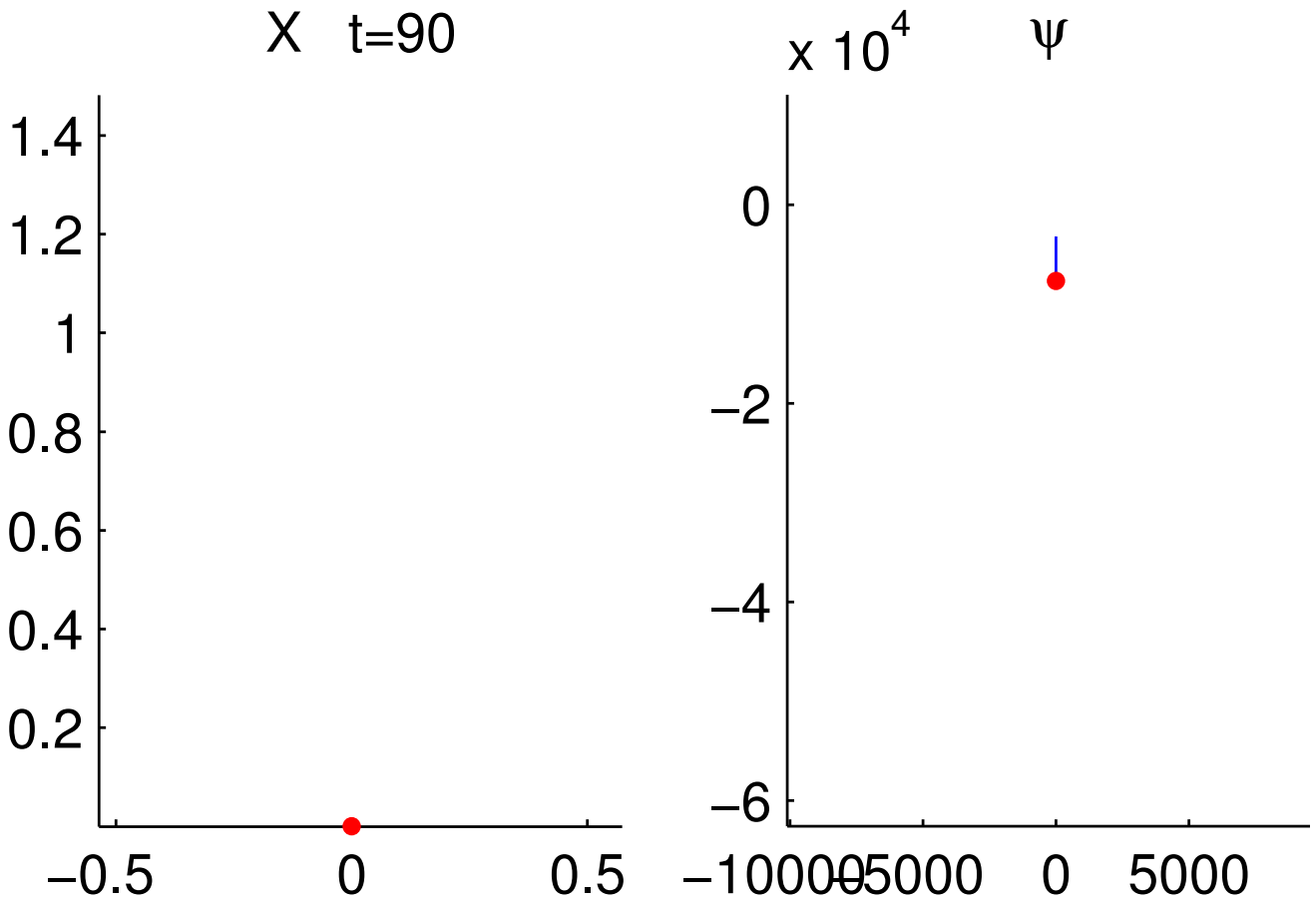
Transition to turbulence $\|X_0\| = 10^{-5.2}$



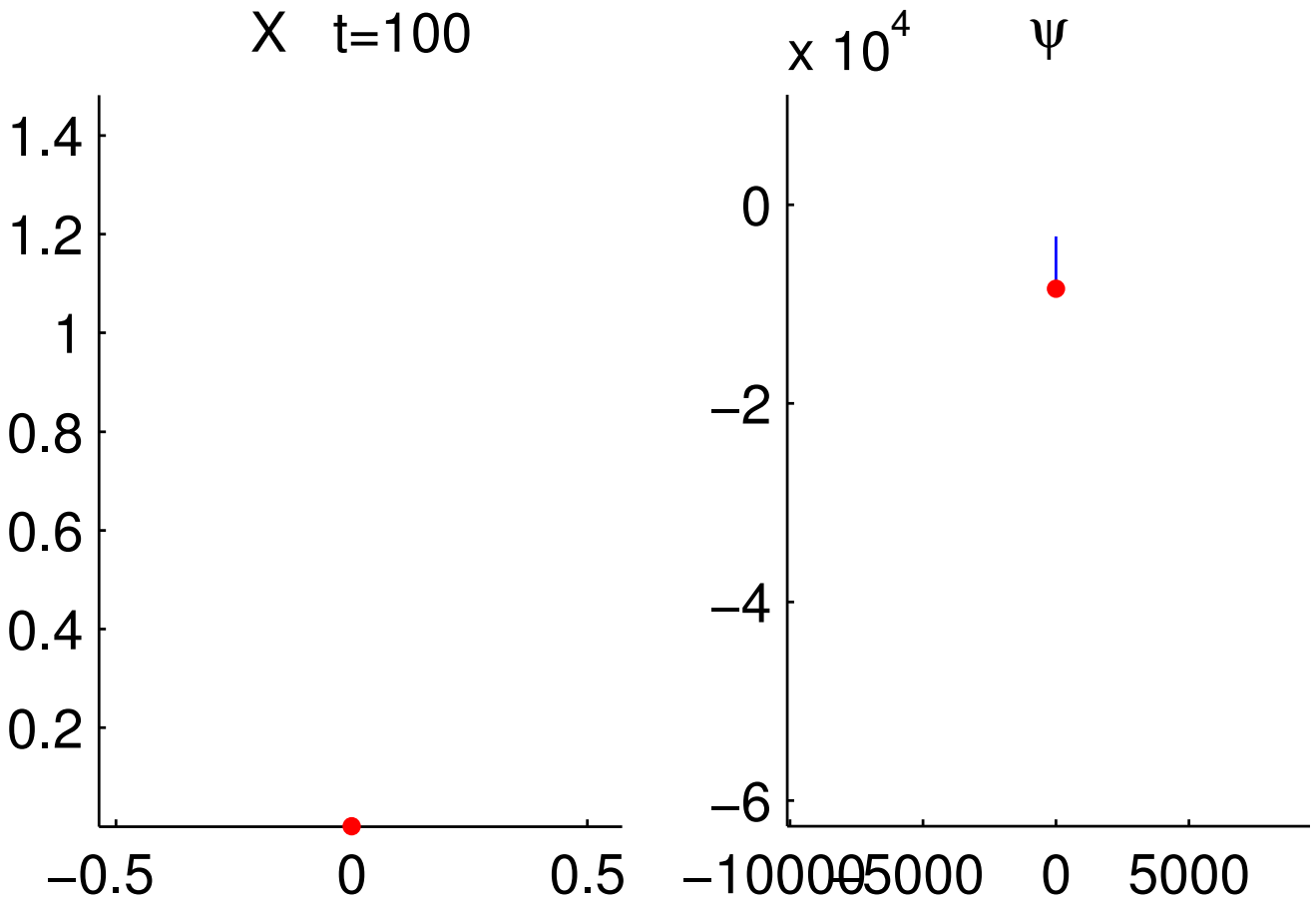
Transition to turbulence $\|X_0\| = 10^{-5.2}$



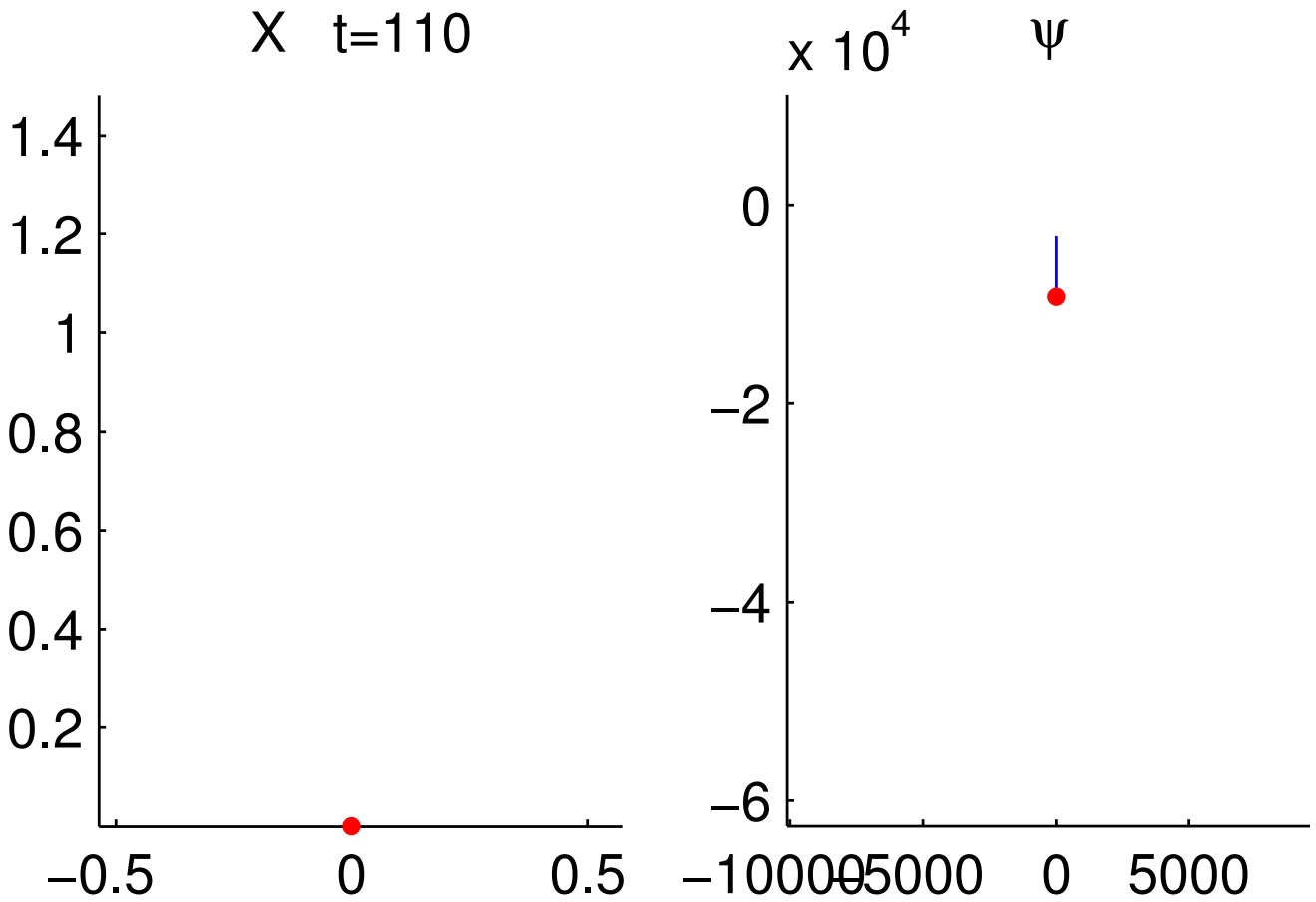
Transition to turbulence $\|X_0\| = 10^{-5.2}$



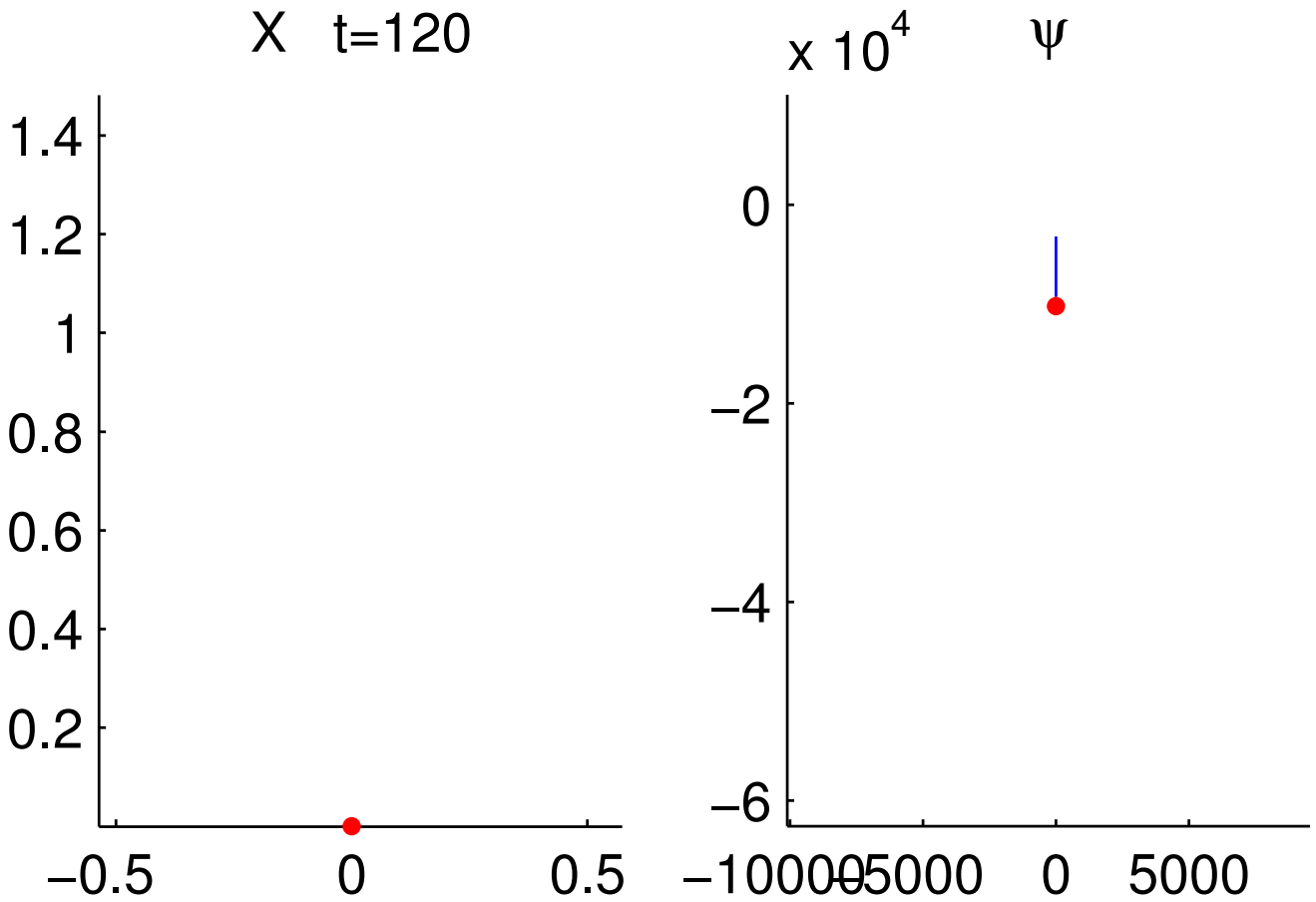
Transition to turbulence $\|X_0\| = 10^{-5.2}$



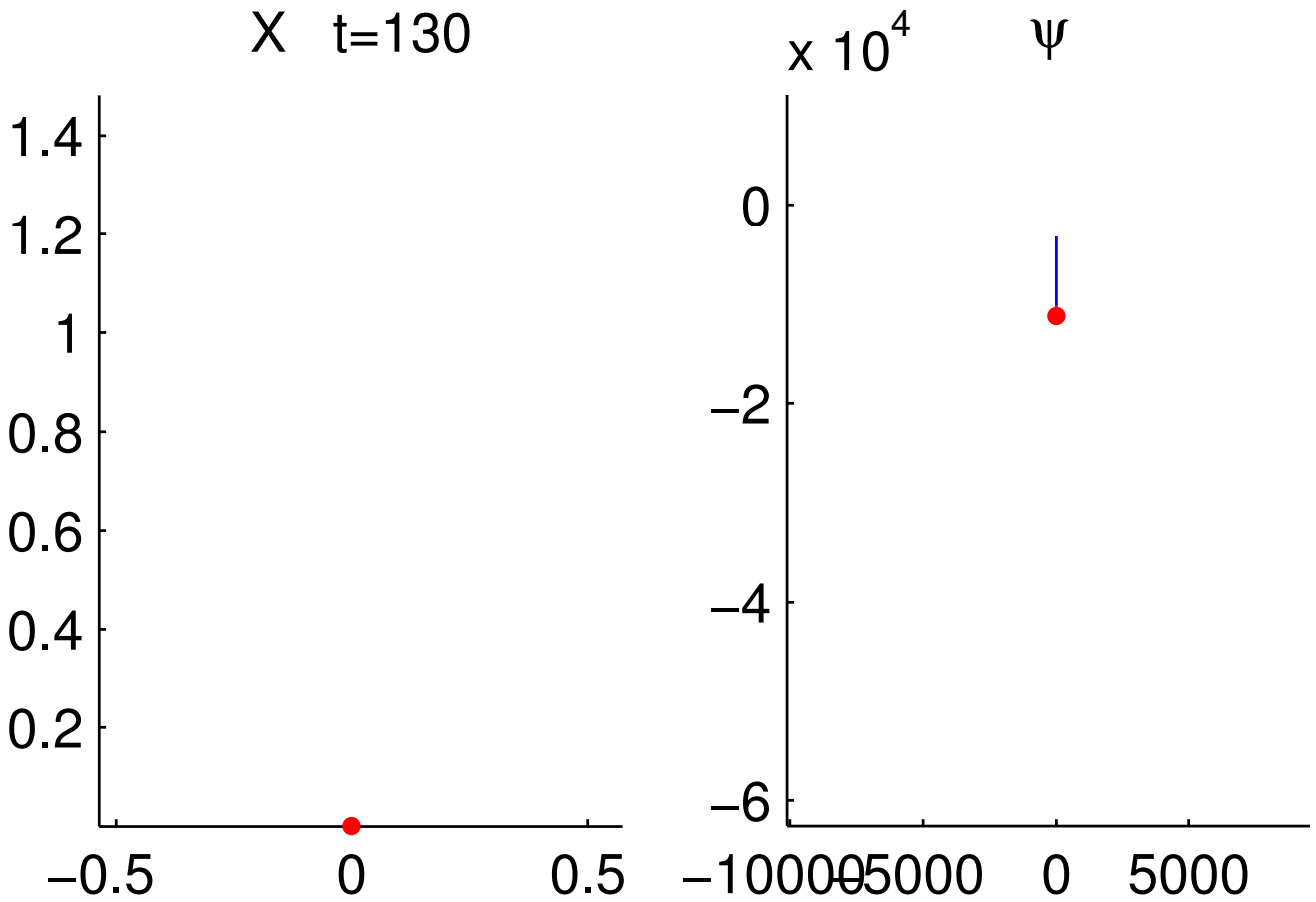
Transition to turbulence $\|X_0\| = 10^{-5.2}$



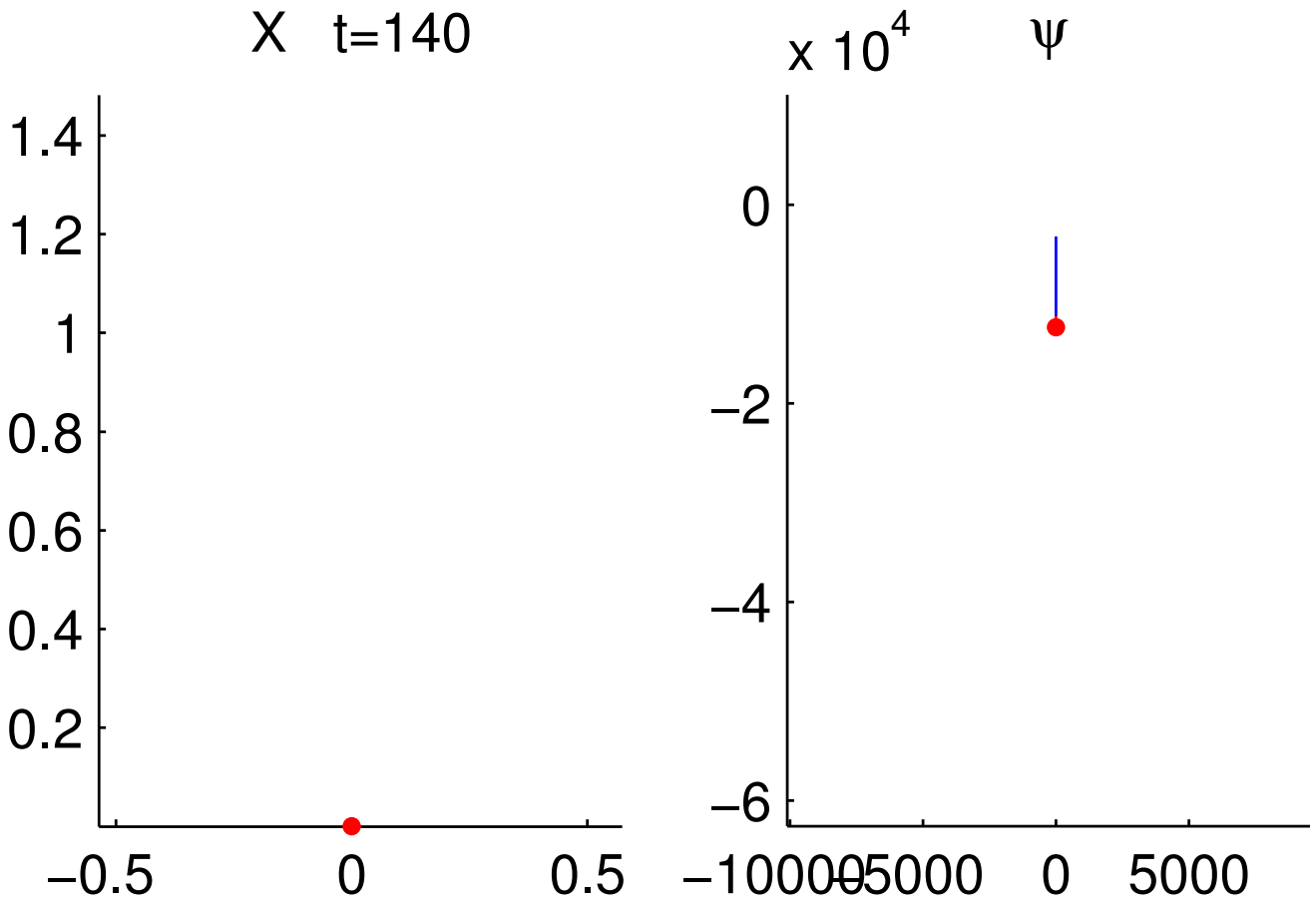
Transition to turbulence $\|X_0\| = 10^{-5.2}$



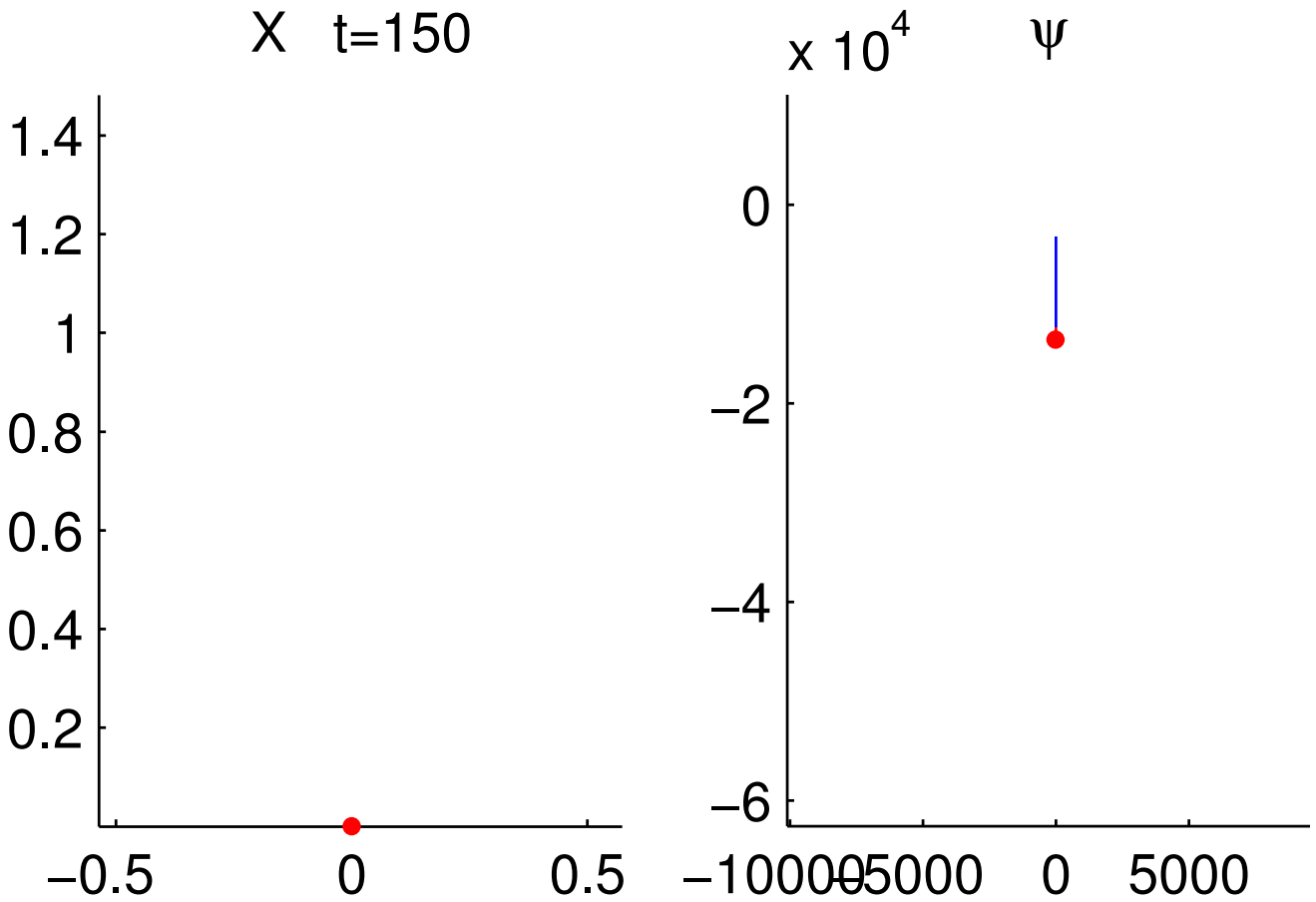
Transition to turbulence $\|X_0\| = 10^{-5.2}$



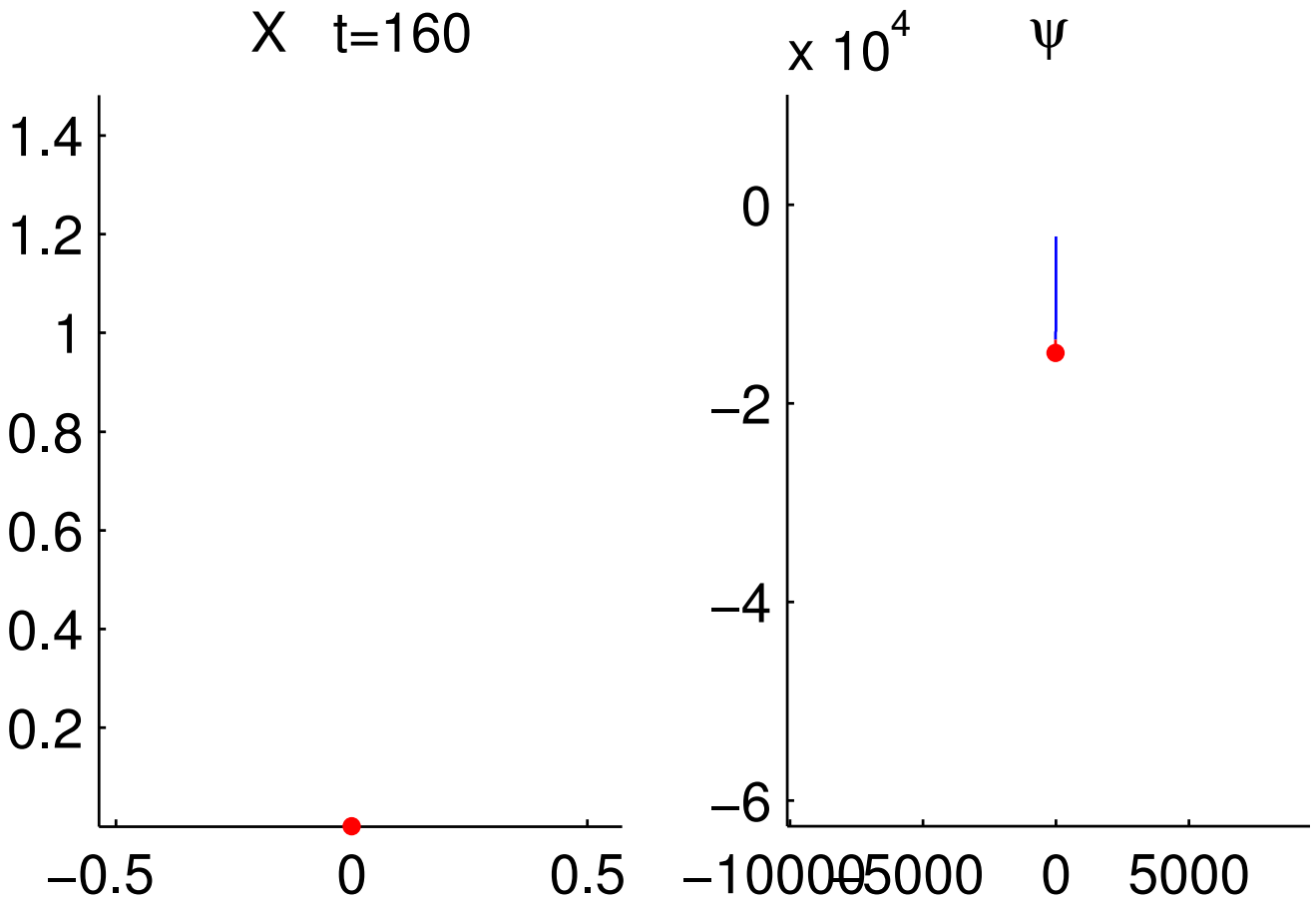
Transition to turbulence $\|X_0\| = 10^{-5.2}$



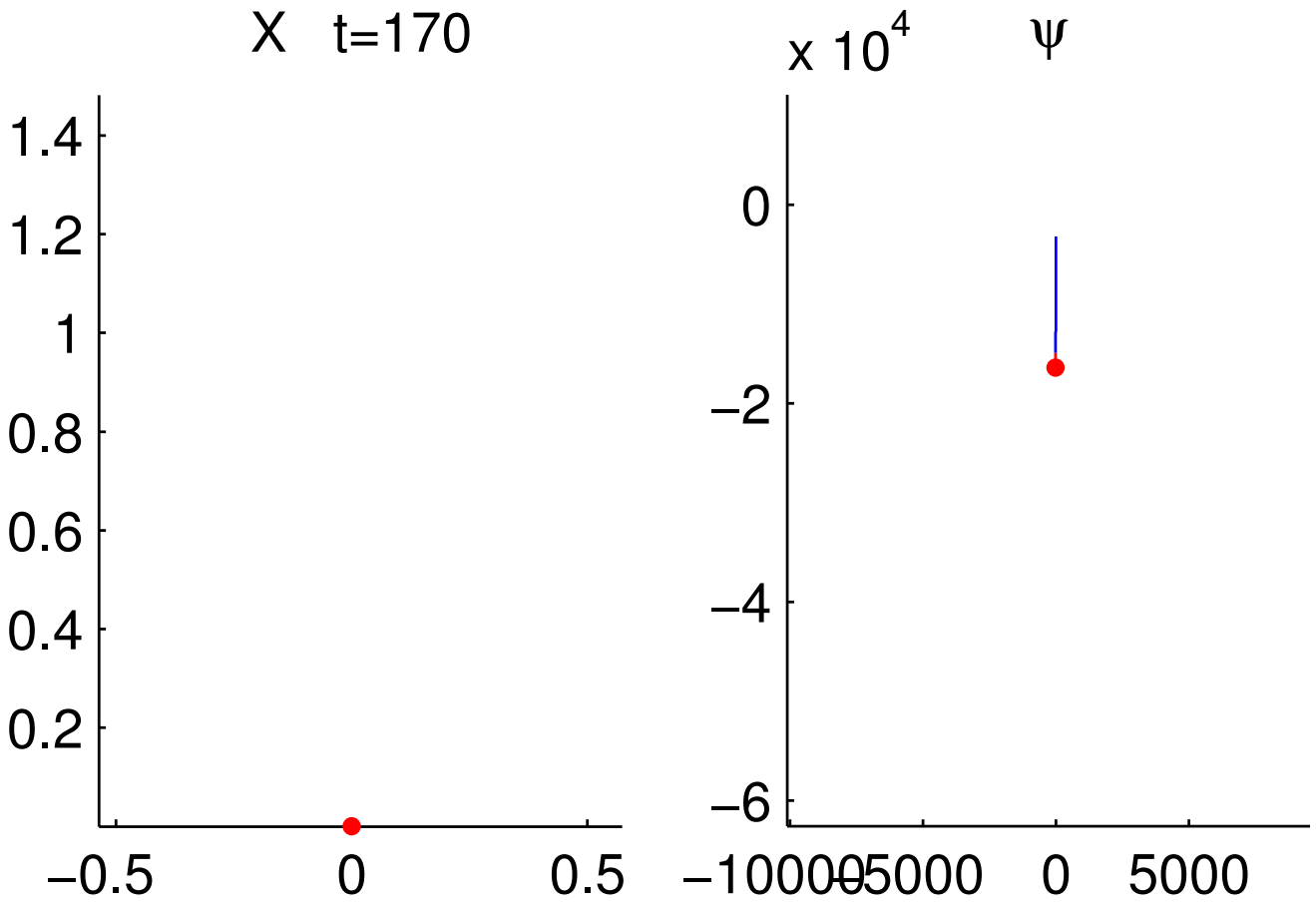
Transition to turbulence $\|X_0\| = 10^{-5.2}$



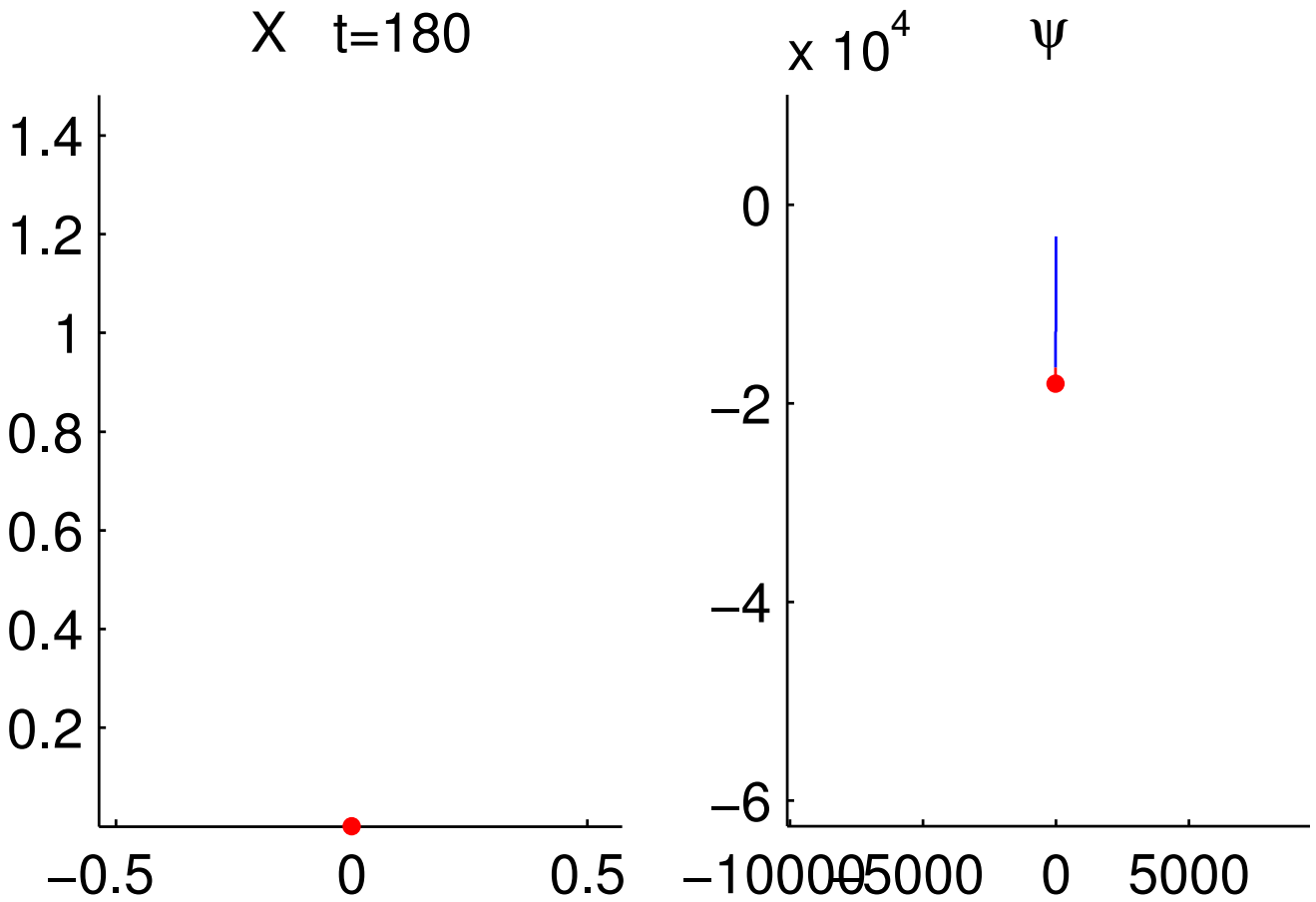
Transition to turbulence $\|X_0\| = 10^{-5.2}$



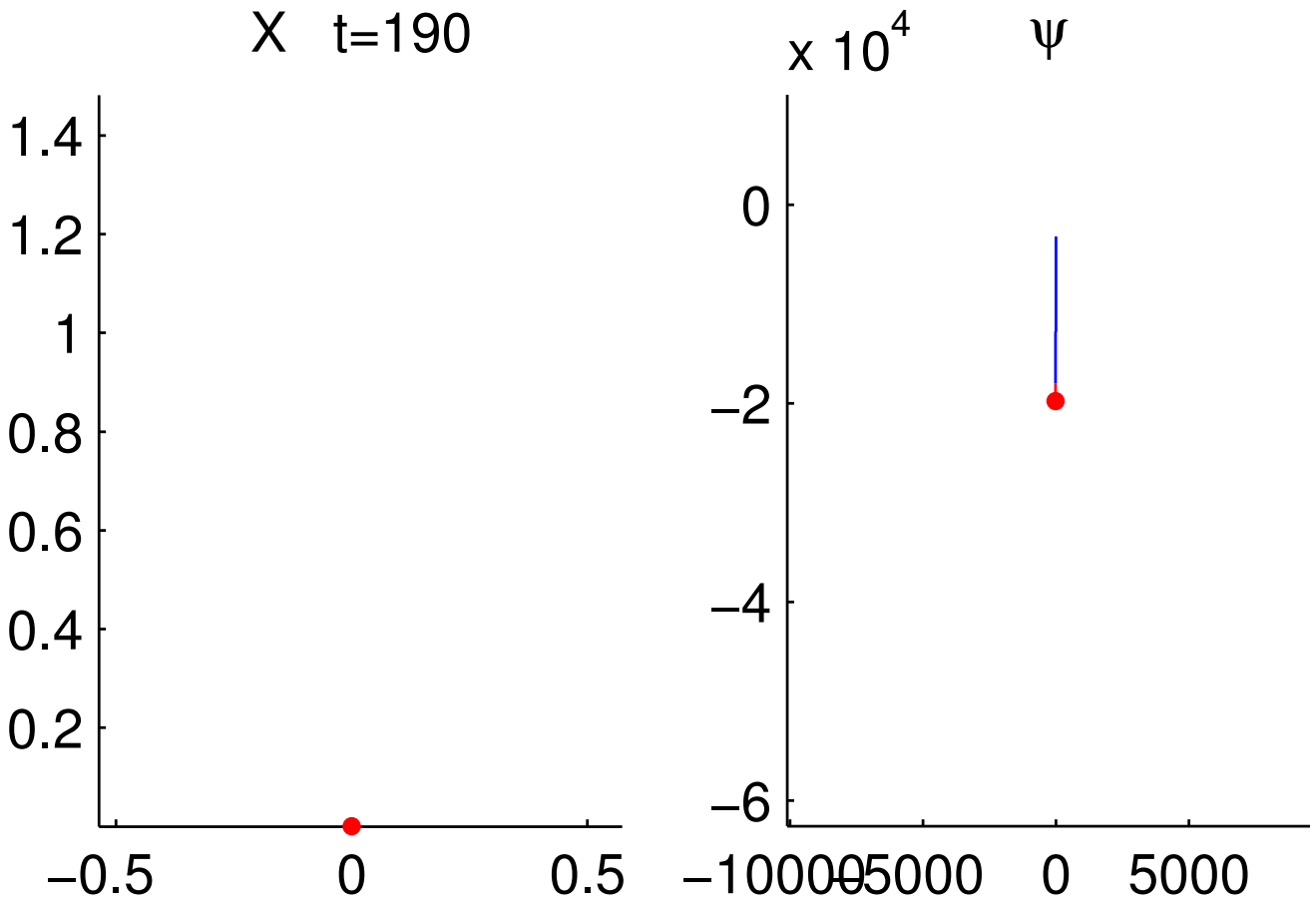
Transition to turbulence $\|X_0\| = 10^{-5.2}$



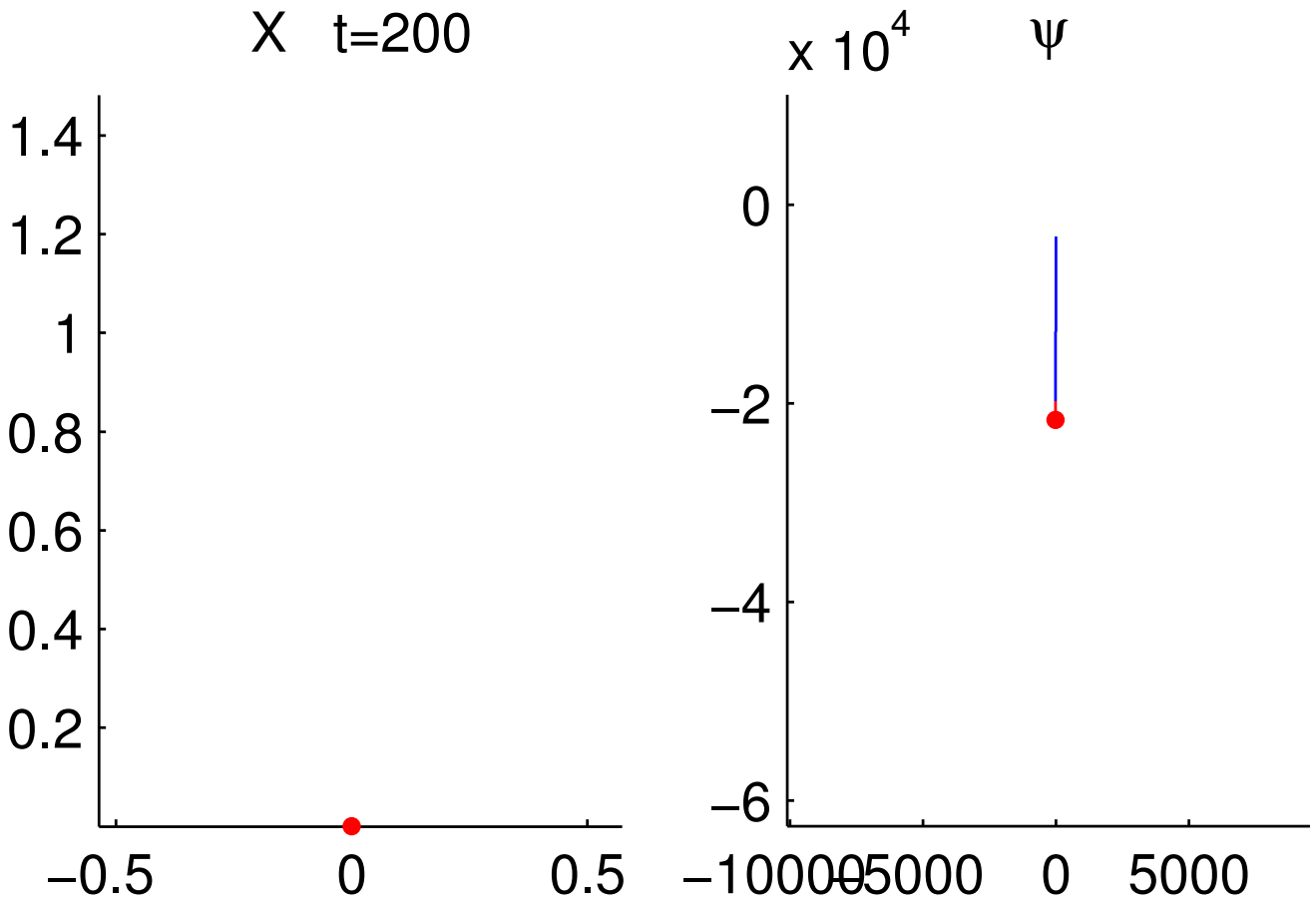
Transition to turbulence $\|X_0\| = 10^{-5.2}$



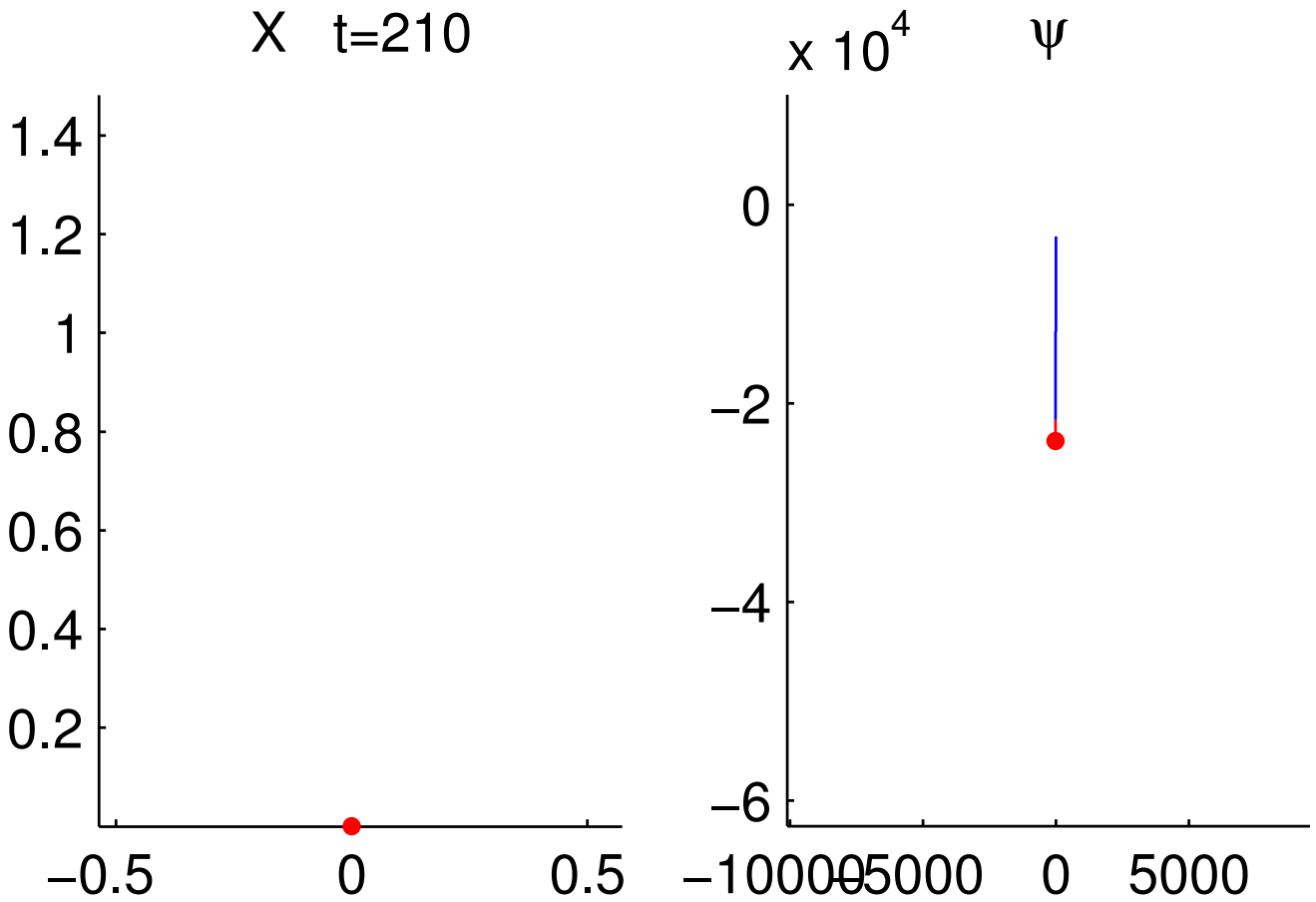
Transition to turbulence $\|X_0\| = 10^{-5.2}$



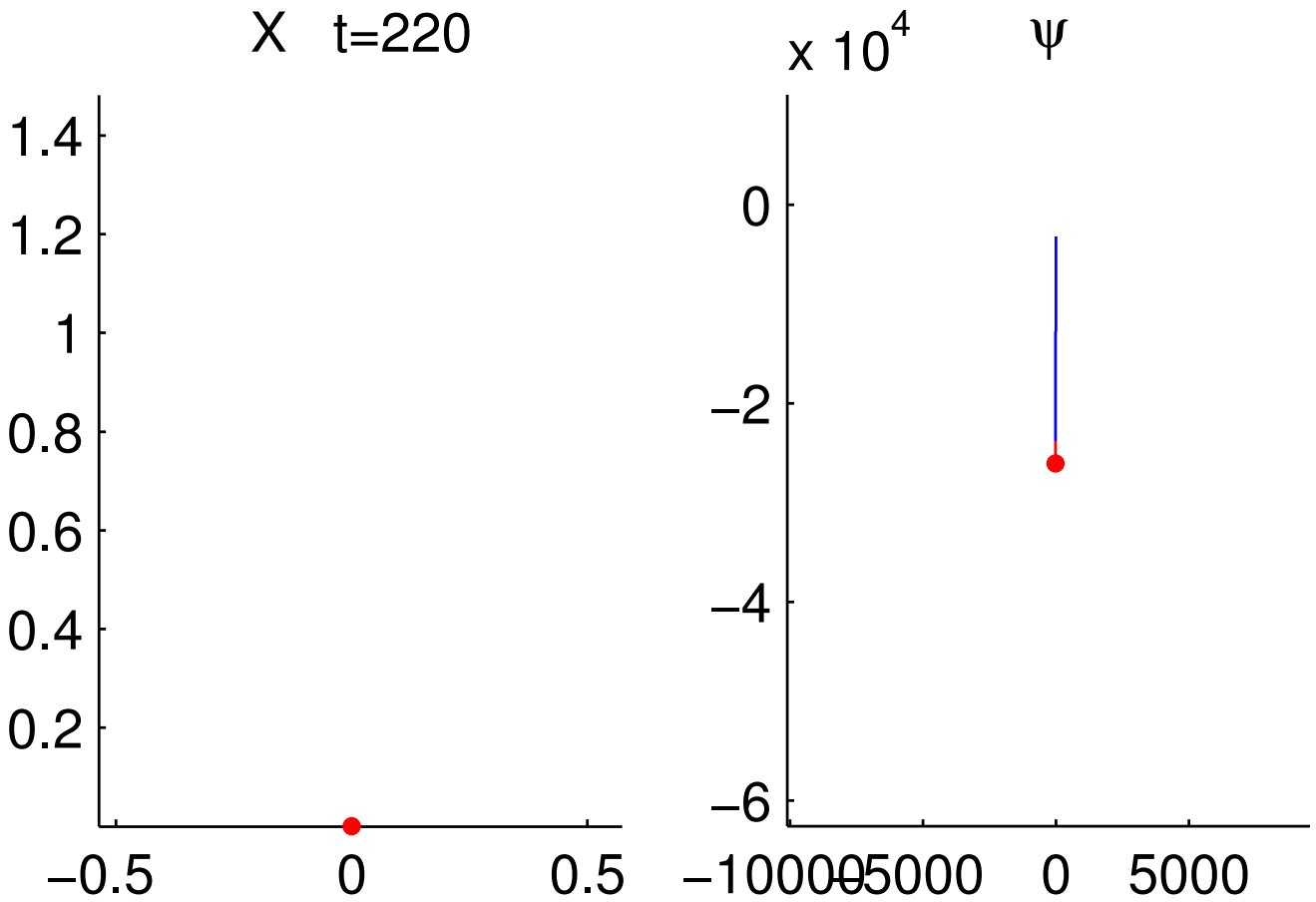
Transition to turbulence $\|X_0\| = 10^{-5.2}$



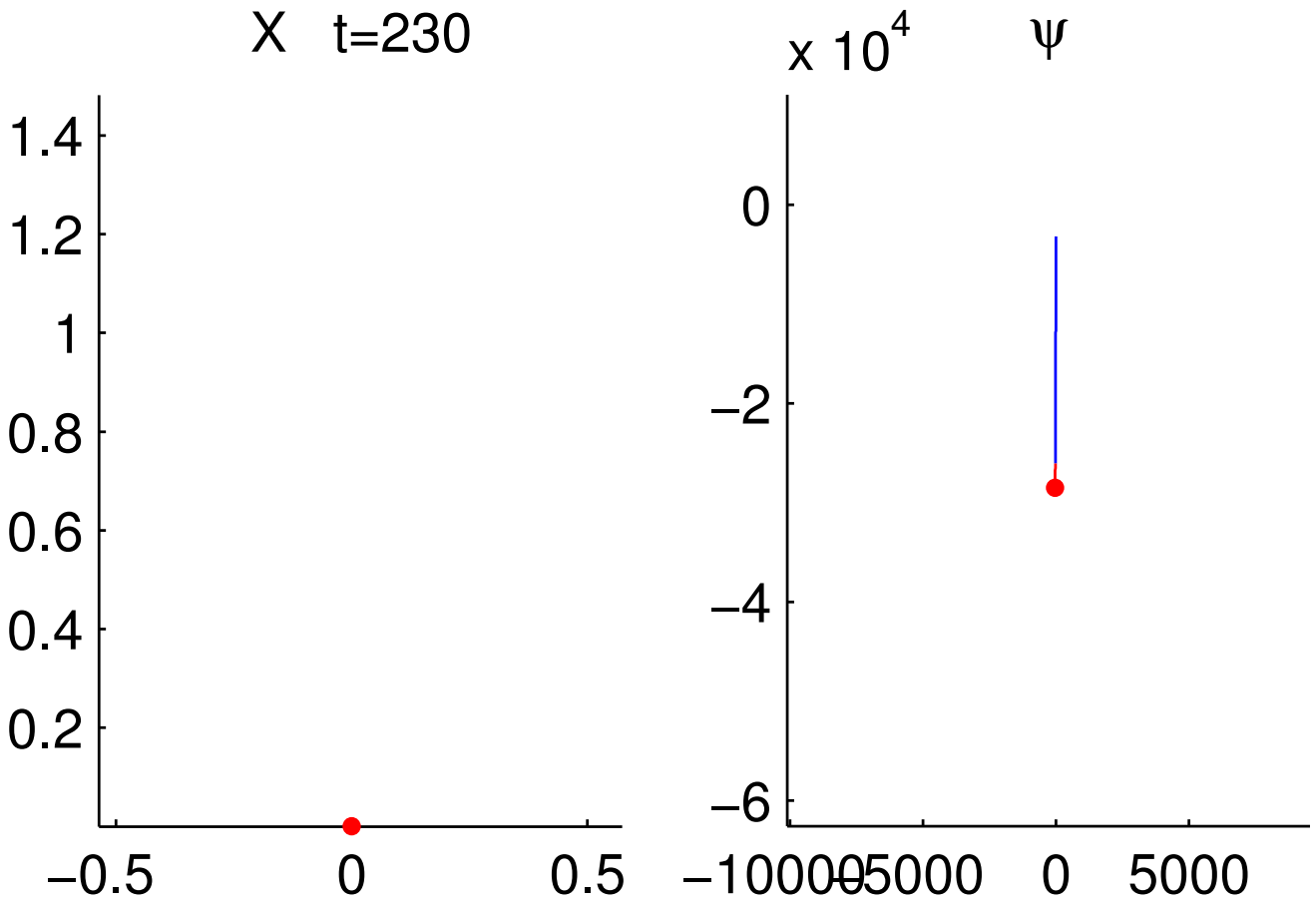
Transition to turbulence $\|X_0\| = 10^{-5.2}$



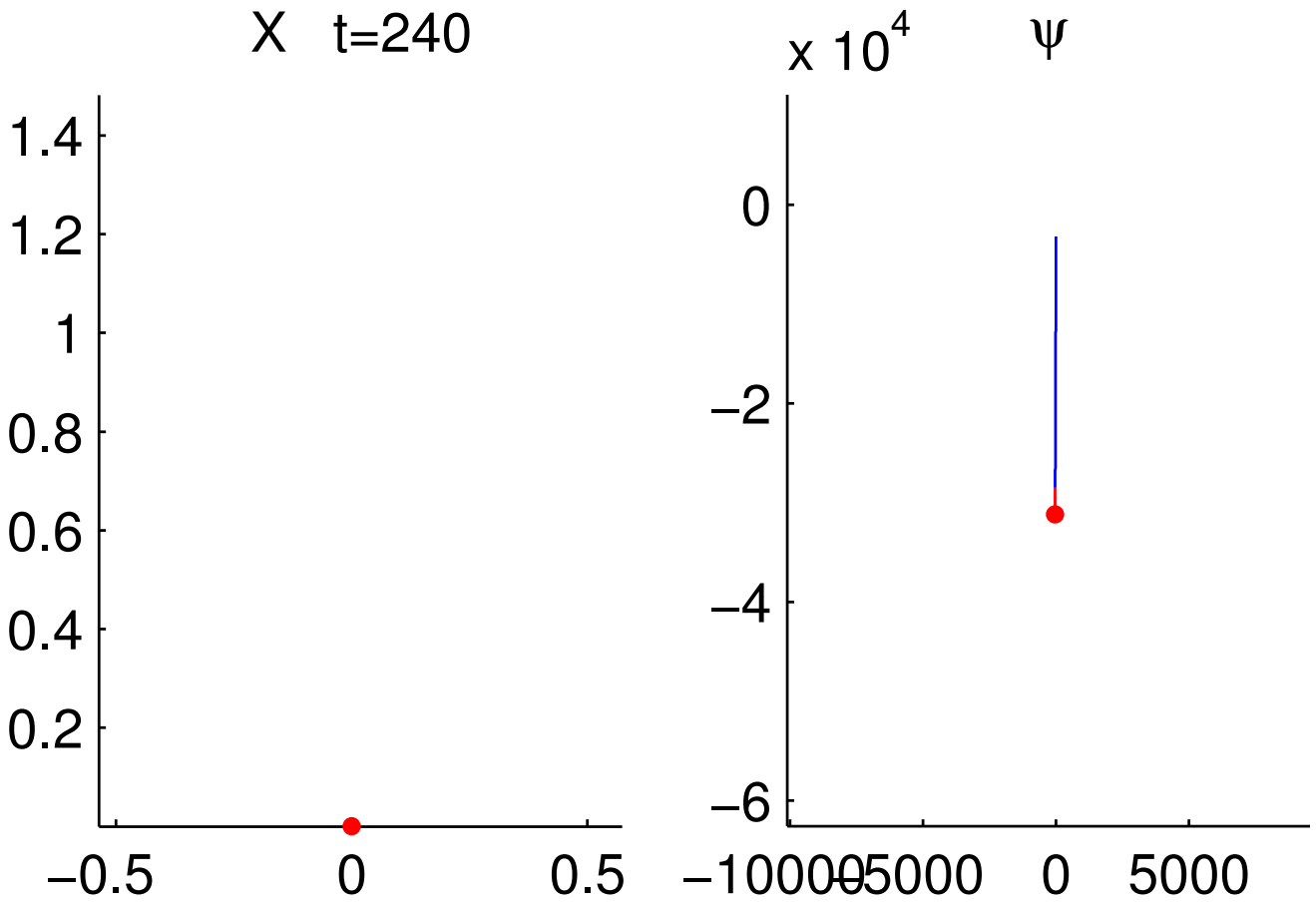
Transition to turbulence $\|X_0\| = 10^{-5.2}$



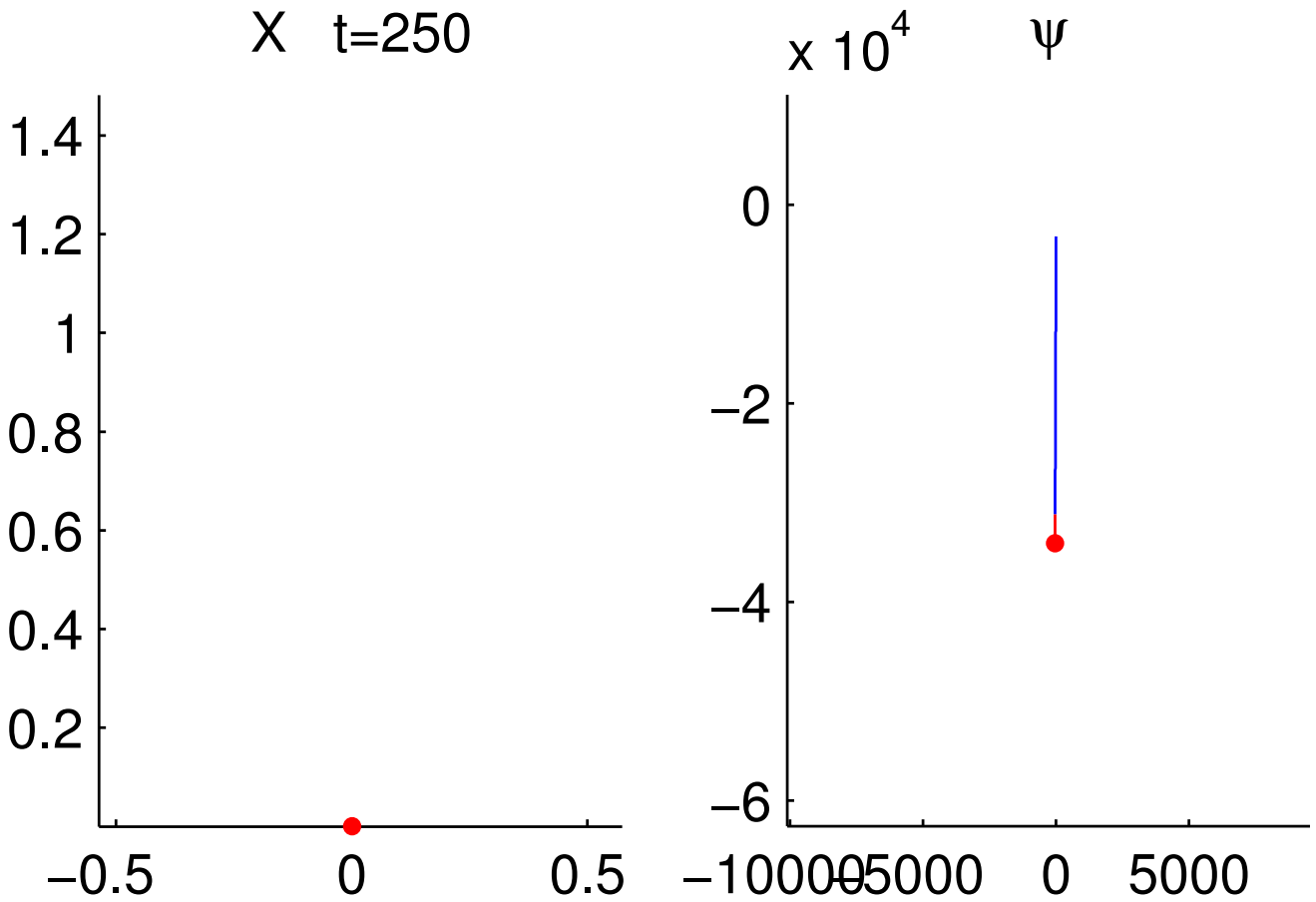
Transition to turbulence $\|X_0\| = 10^{-5.2}$



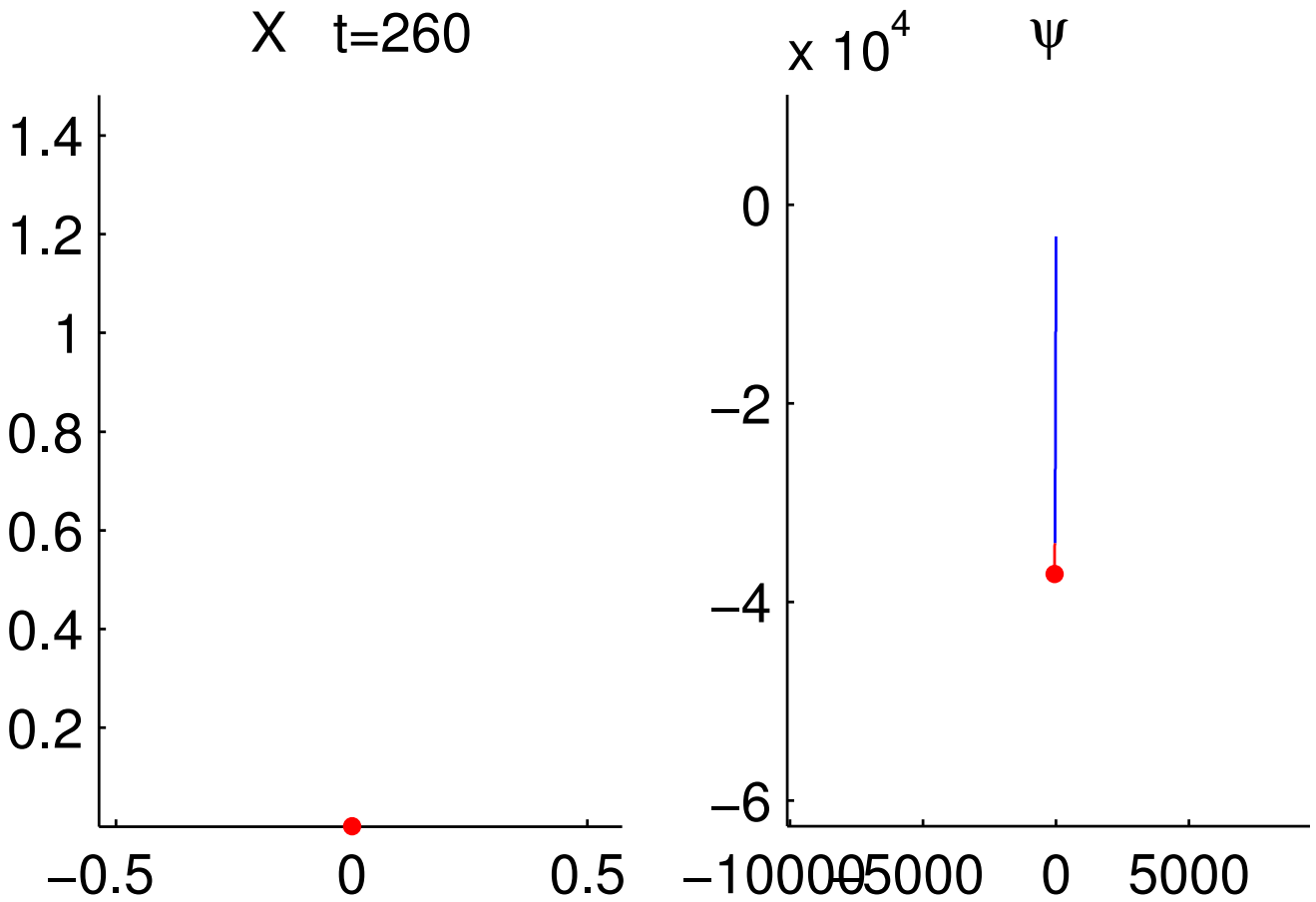
Transition to turbulence $\|X_0\| = 10^{-5.2}$



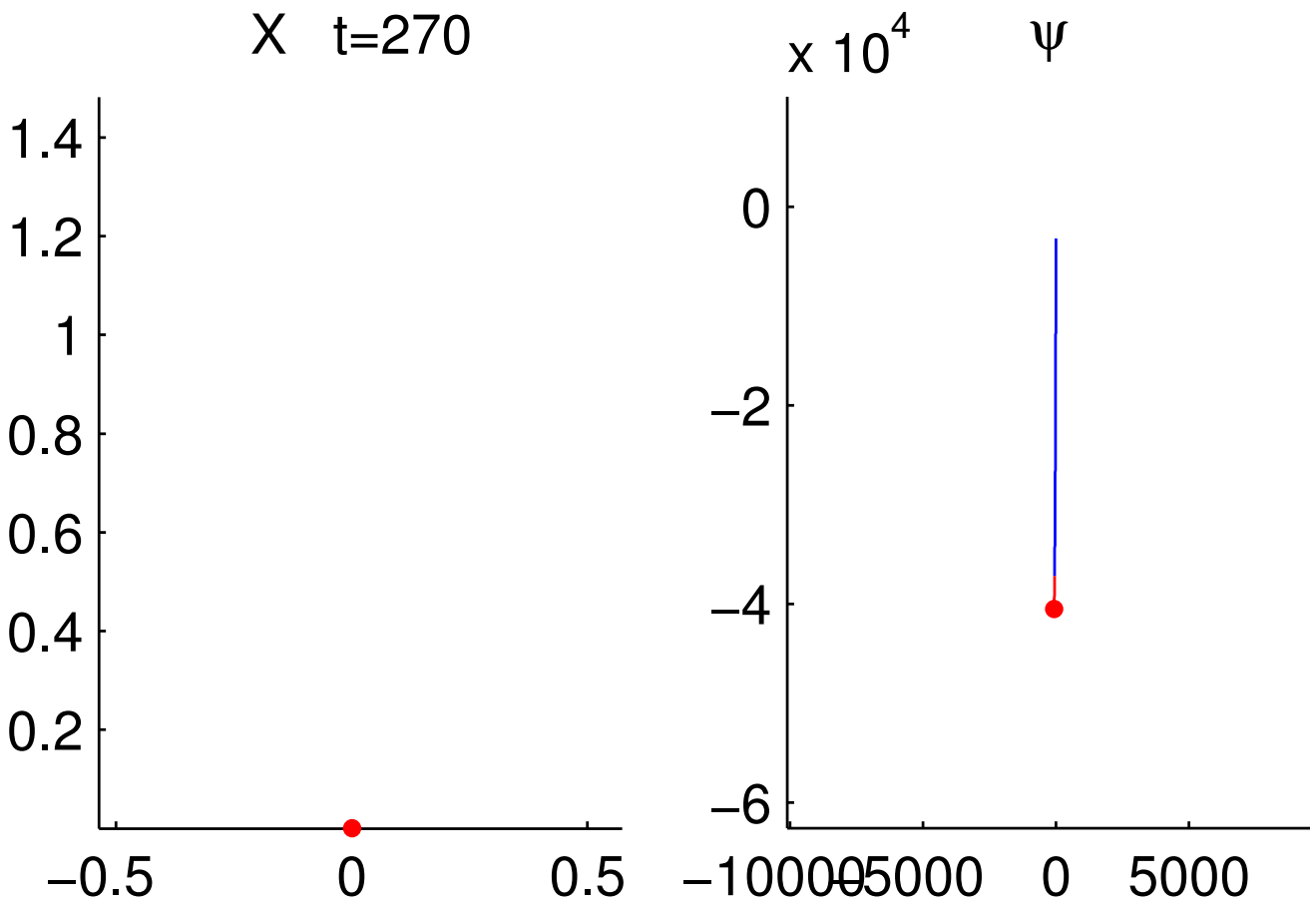
Transition to turbulence $\|X_0\| = 10^{-5.2}$



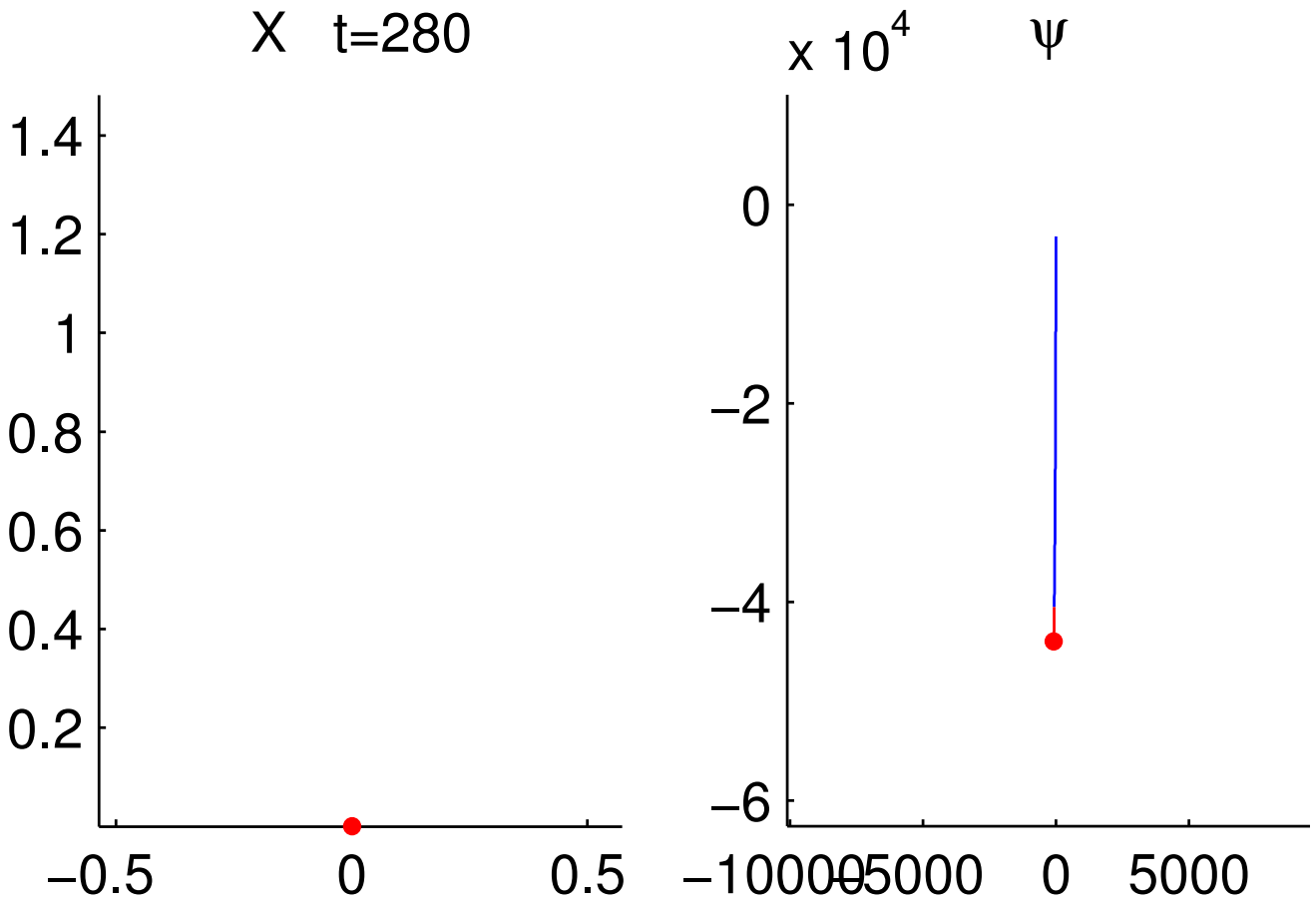
Transition to turbulence $\|X_0\| = 10^{-5.2}$



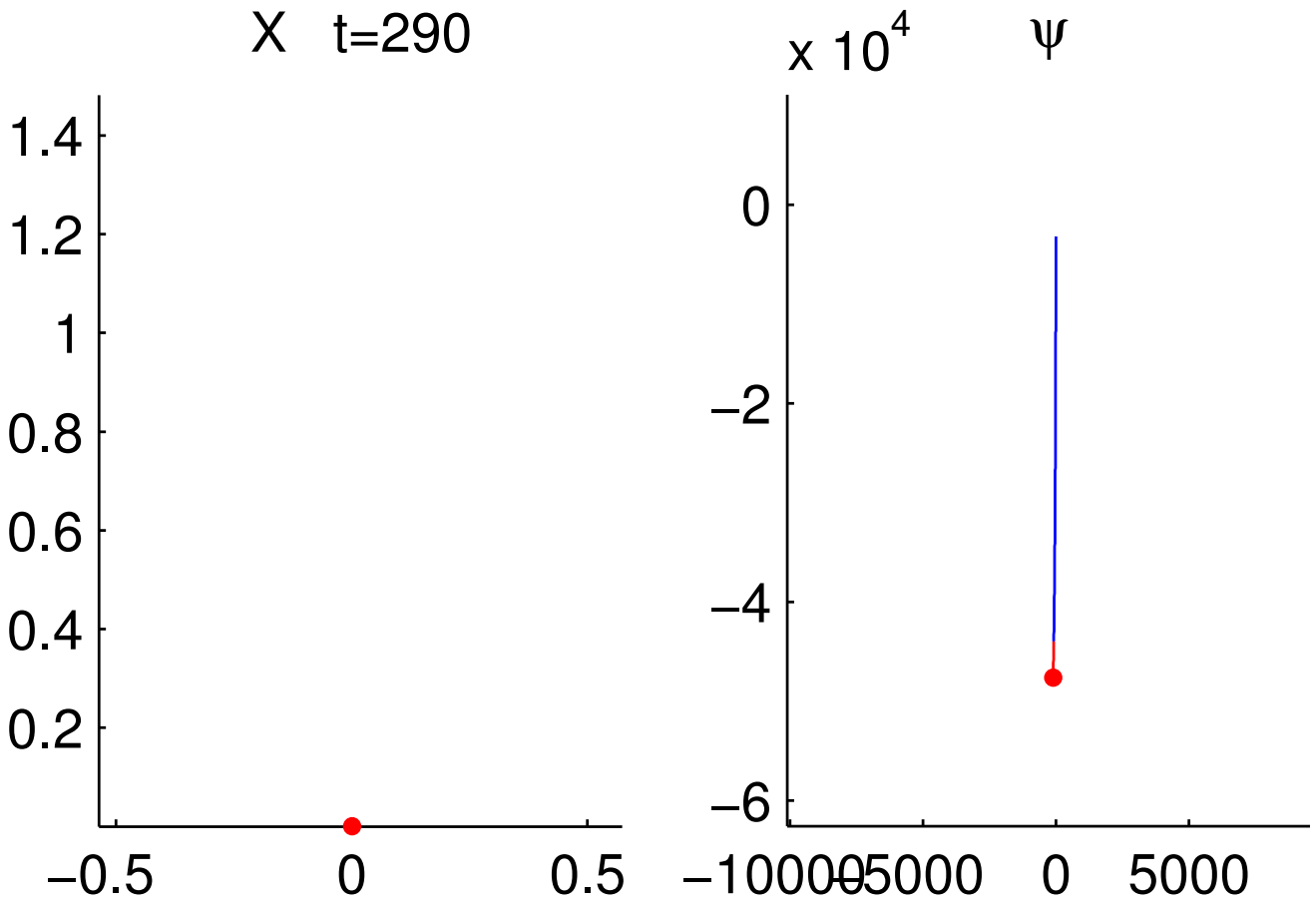
Transition to turbulence $\|X_0\| = 10^{-5.2}$



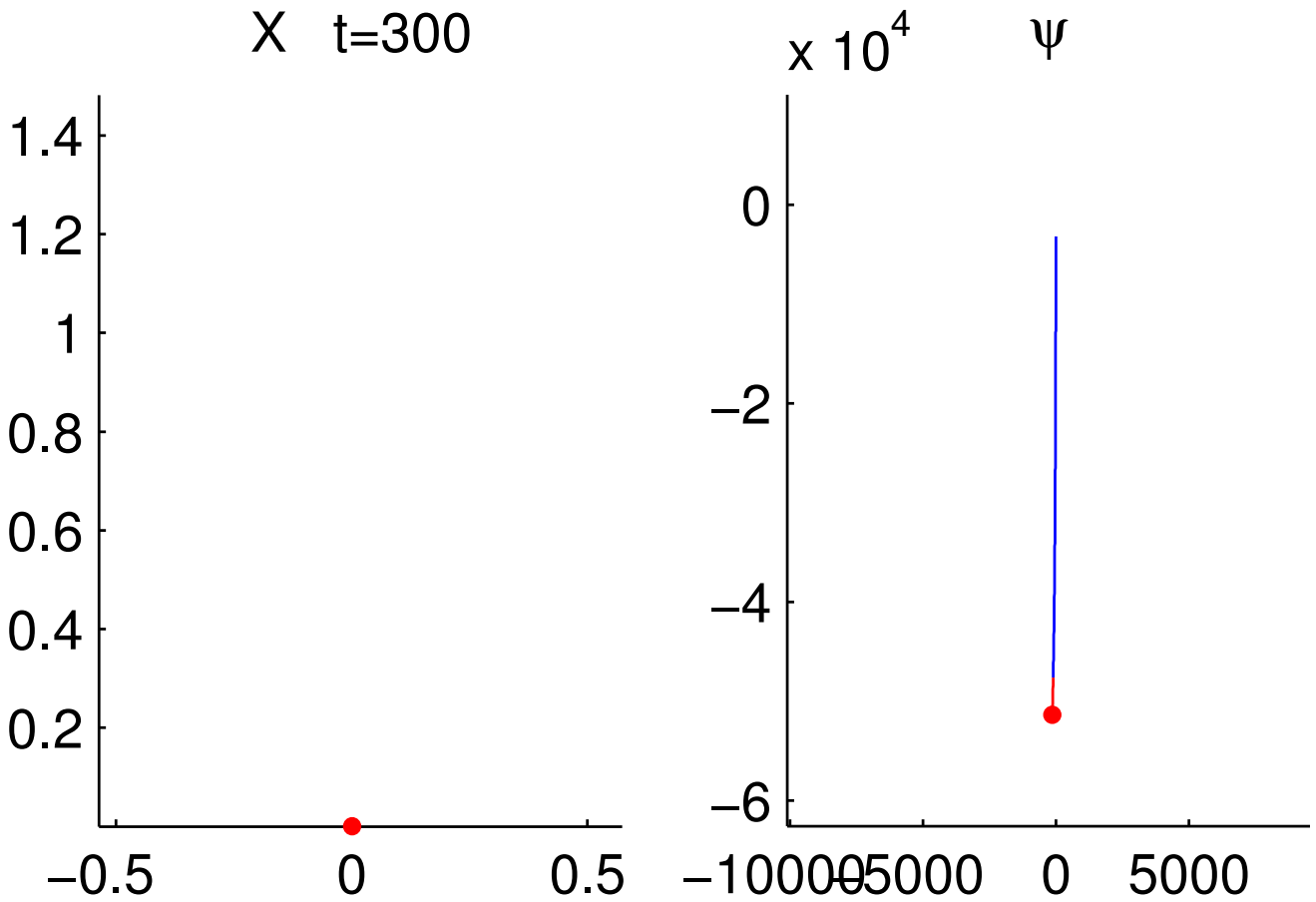
Transition to turbulence $\|X_0\| = 10^{-5.2}$



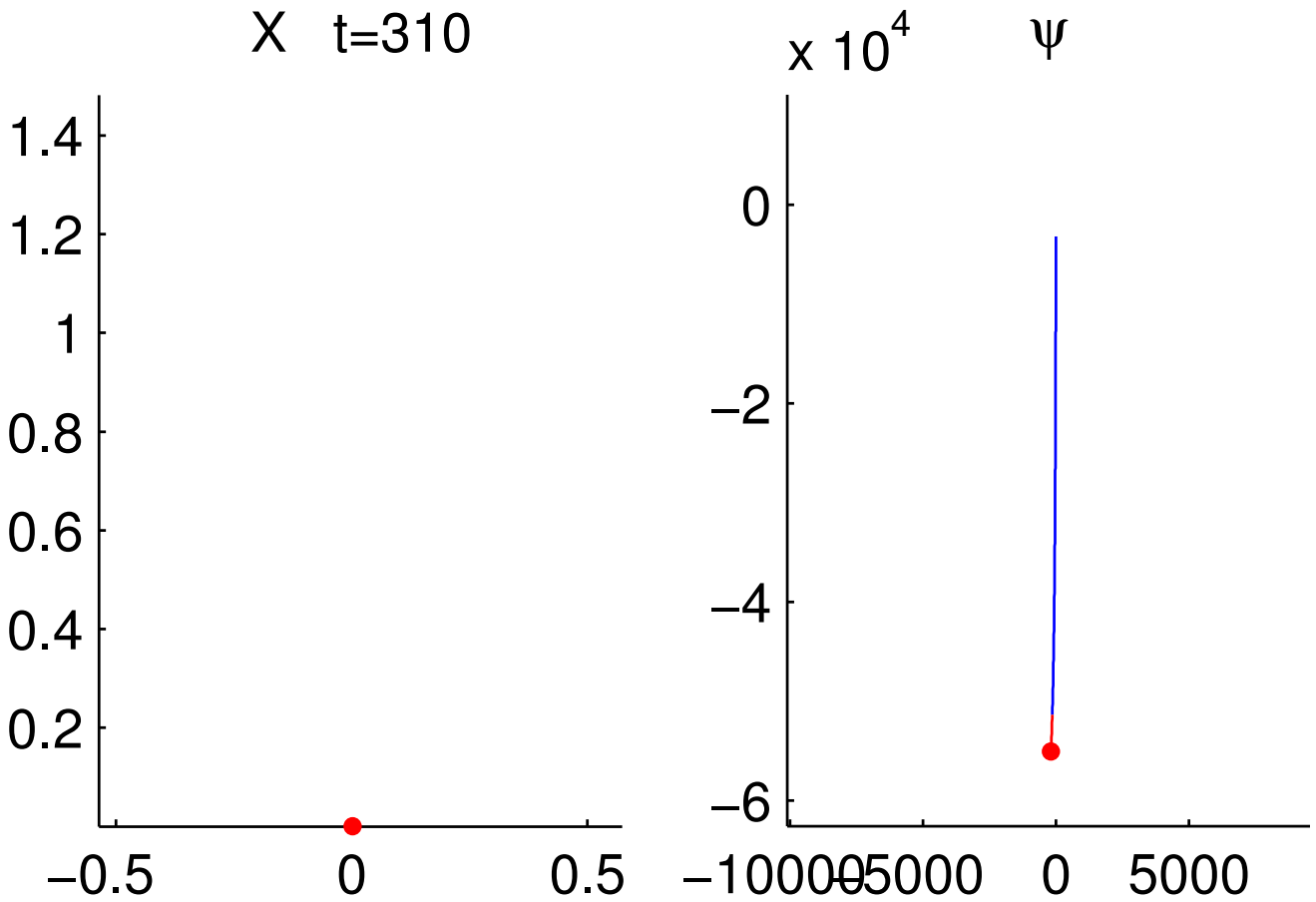
Transition to turbulence $\|X_0\| = 10^{-5.2}$



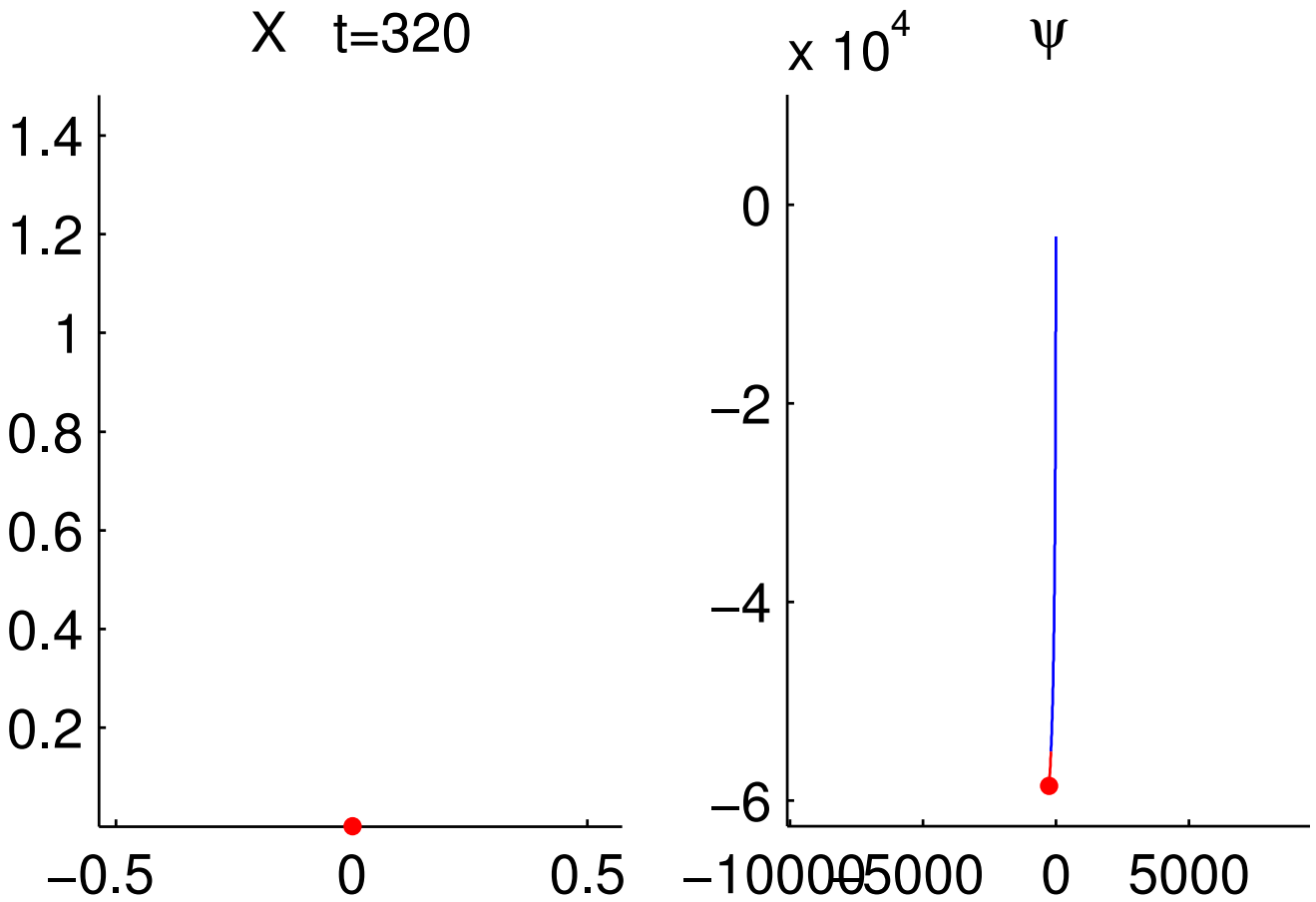
Transition to turbulence $\|X_0\| = 10^{-5.2}$



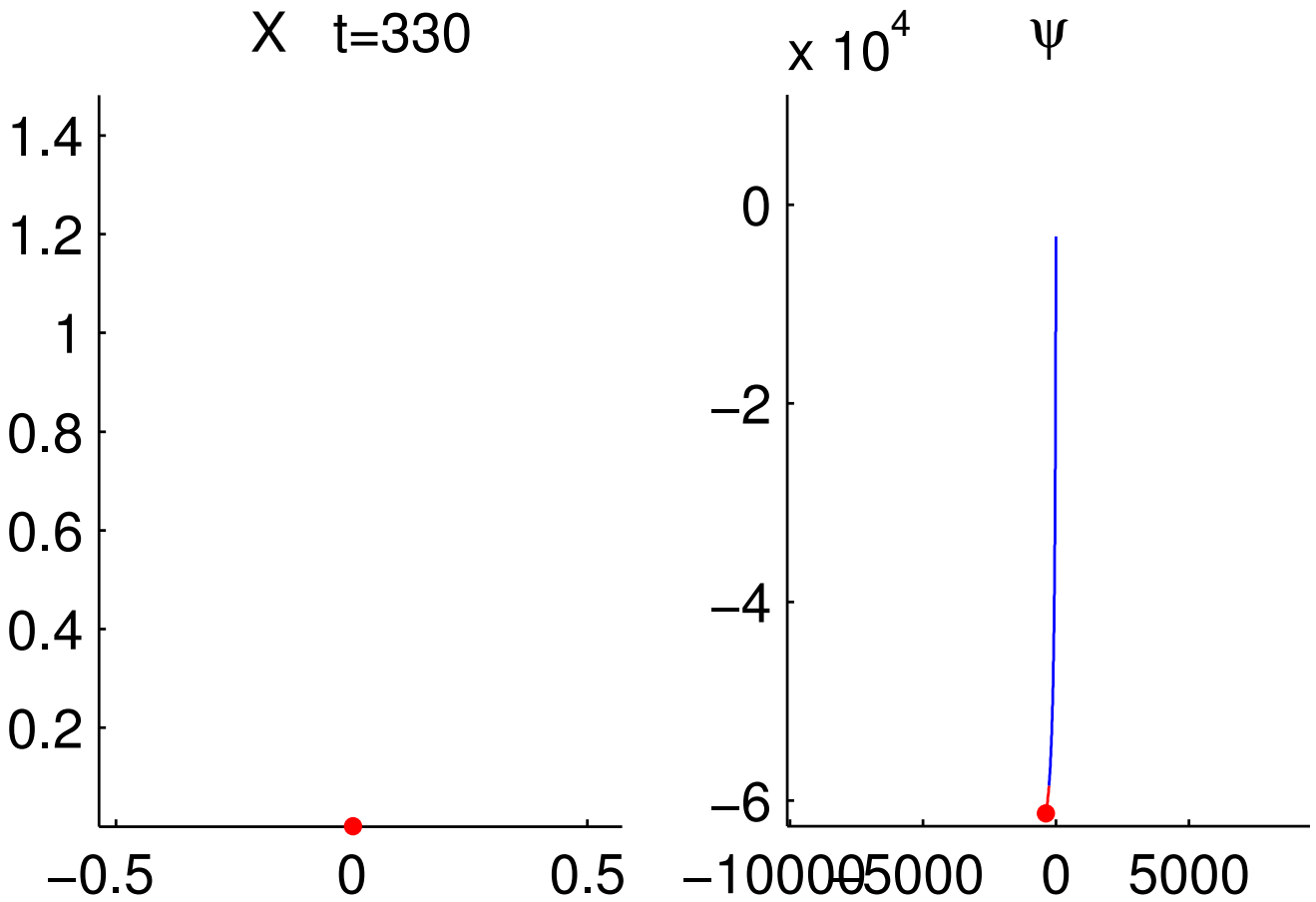
Transition to turbulence $\|X_0\| = 10^{-5.2}$



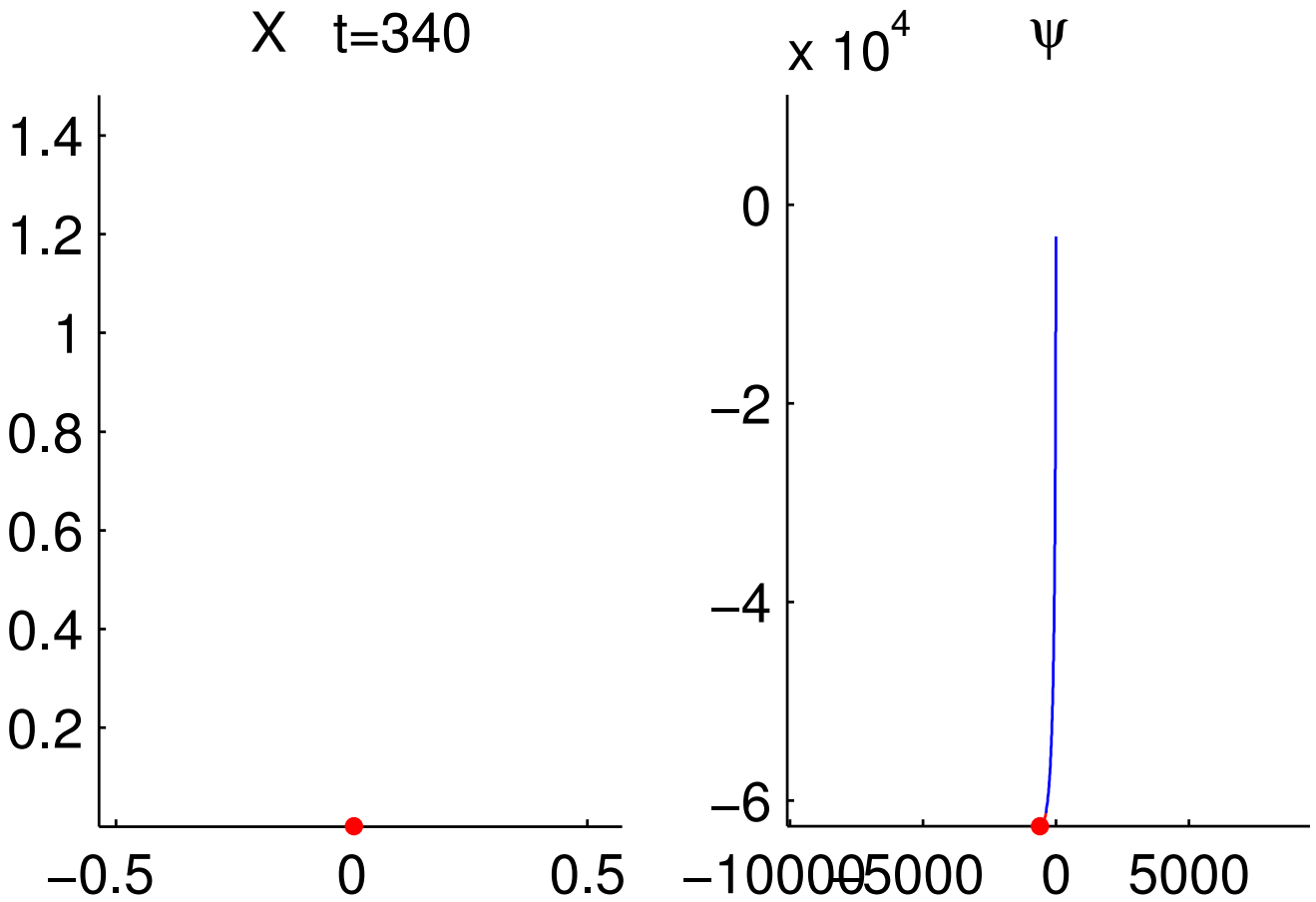
Transition to turbulence $\|X_0\| = 10^{-5.2}$



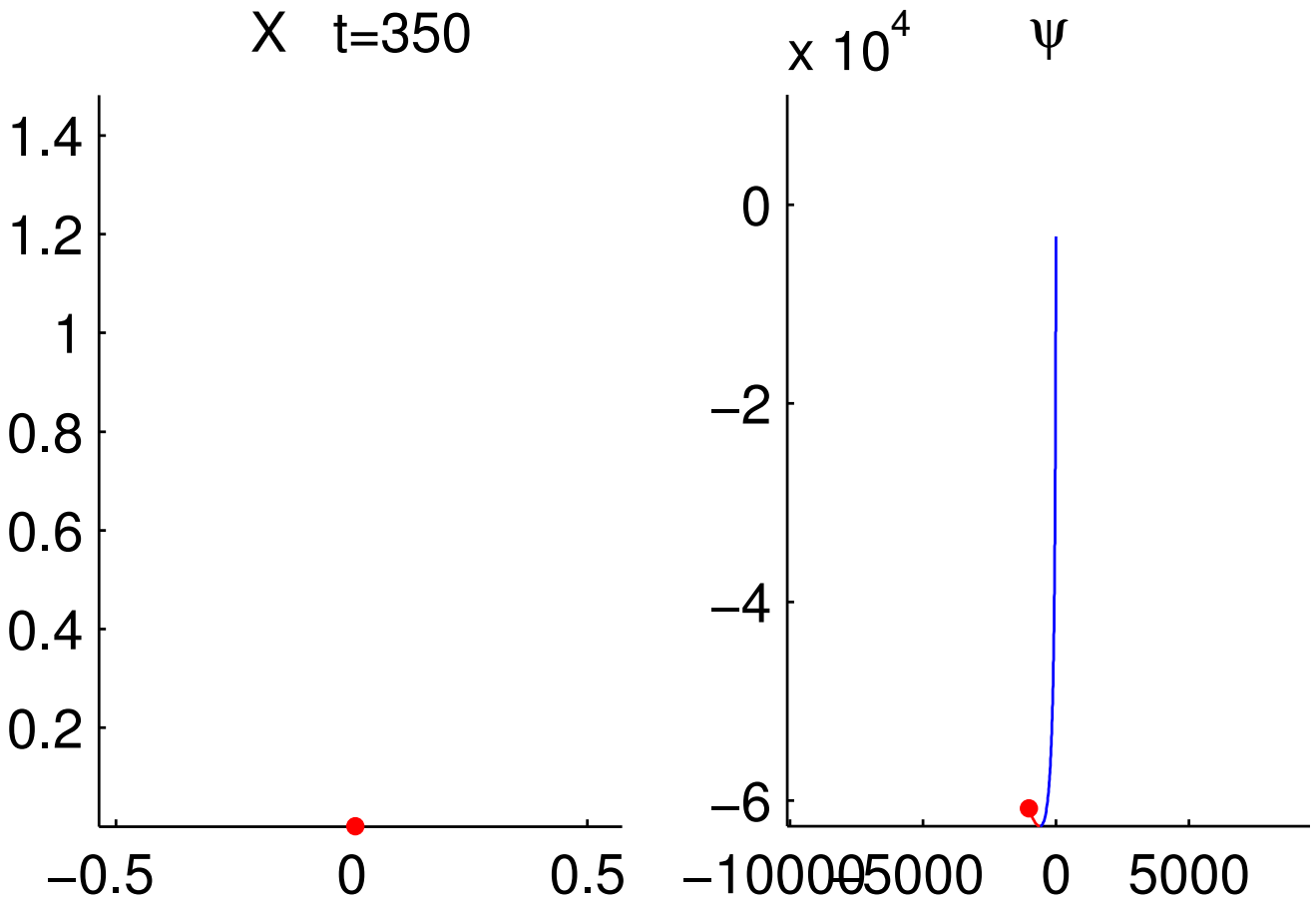
Transition to turbulence $\|X_0\| = 10^{-5.2}$



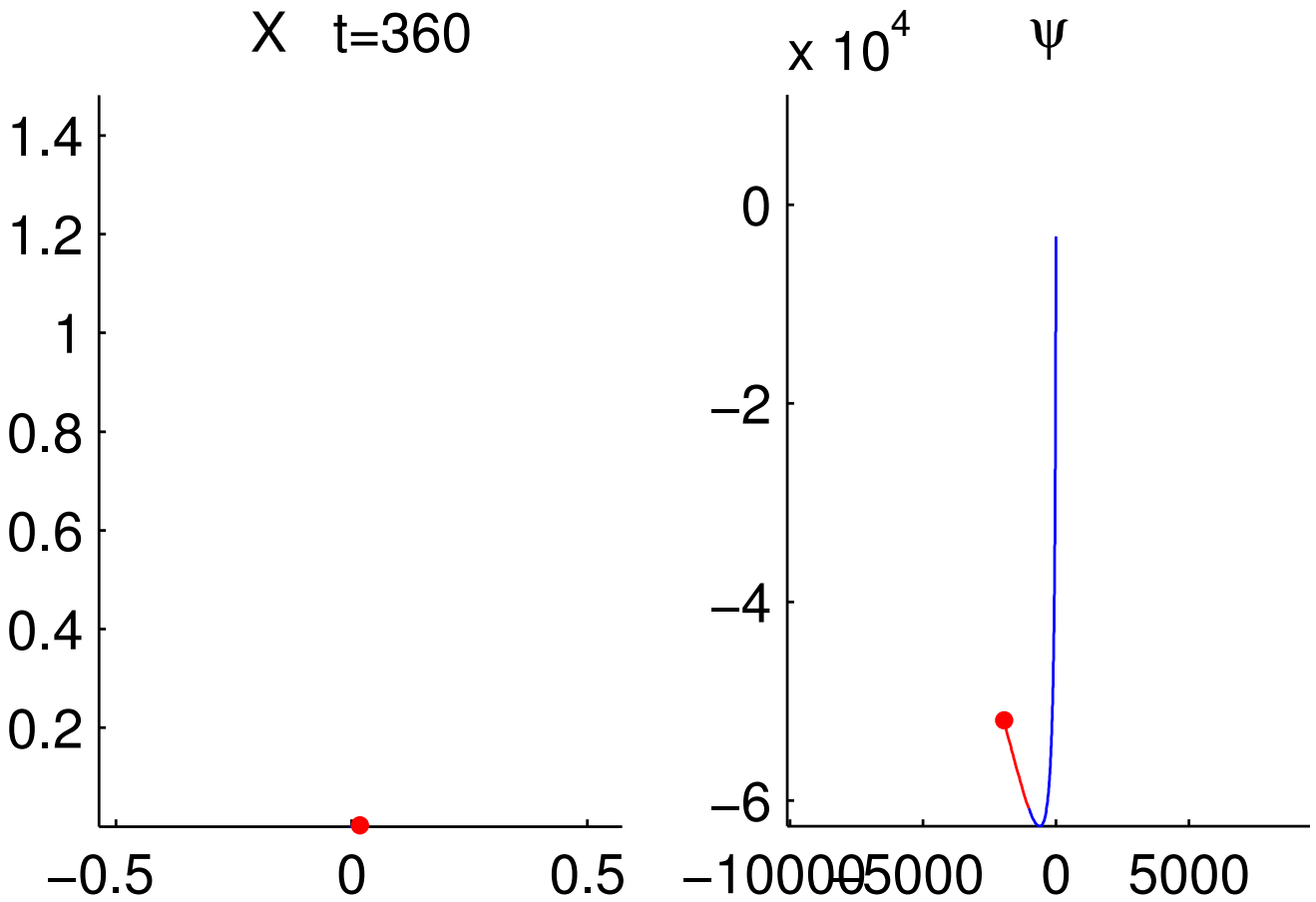
Transition to turbulence $\|X_0\| = 10^{-5.2}$



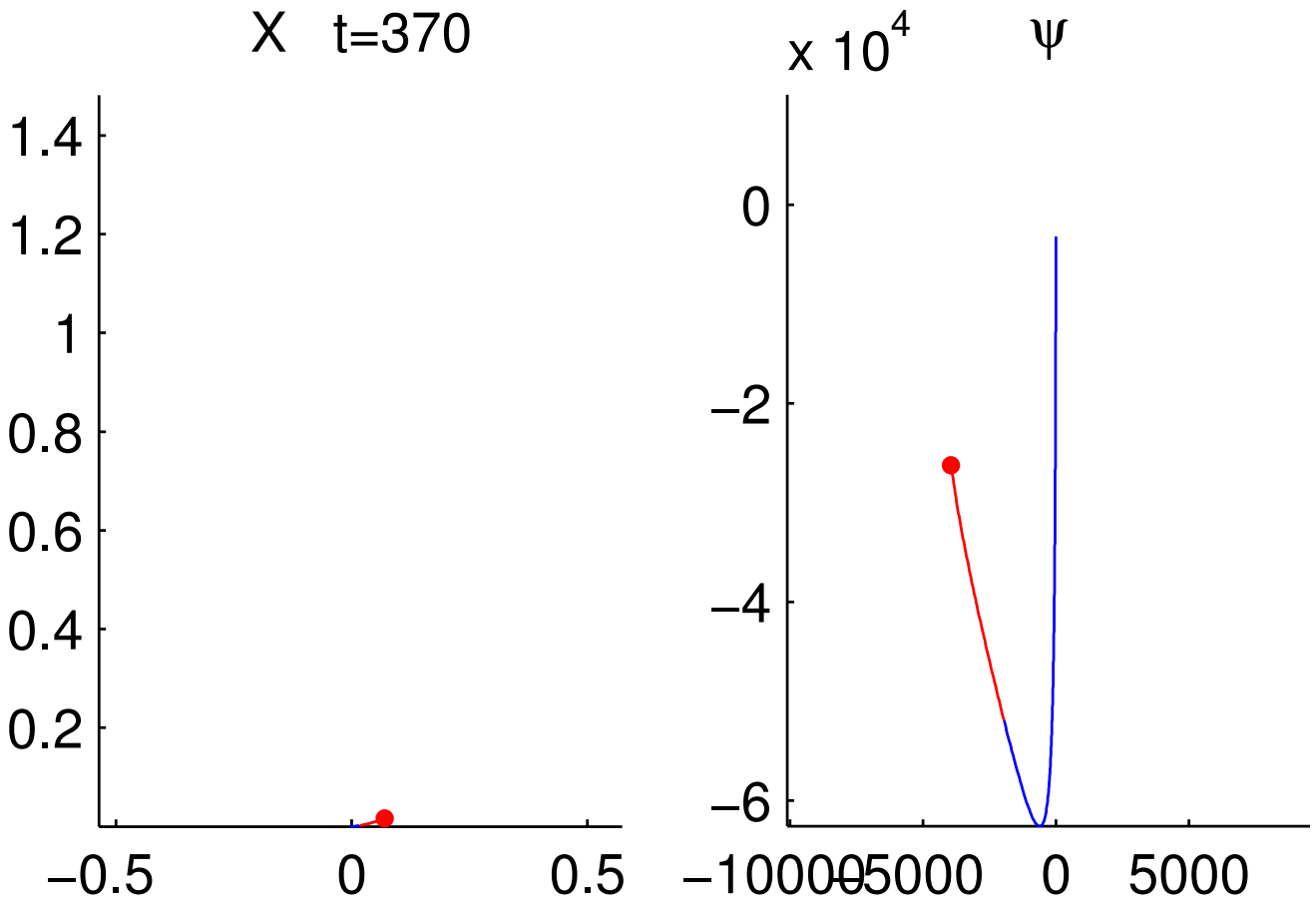
Transition to turbulence $\|X_0\| = 10^{-5.2}$



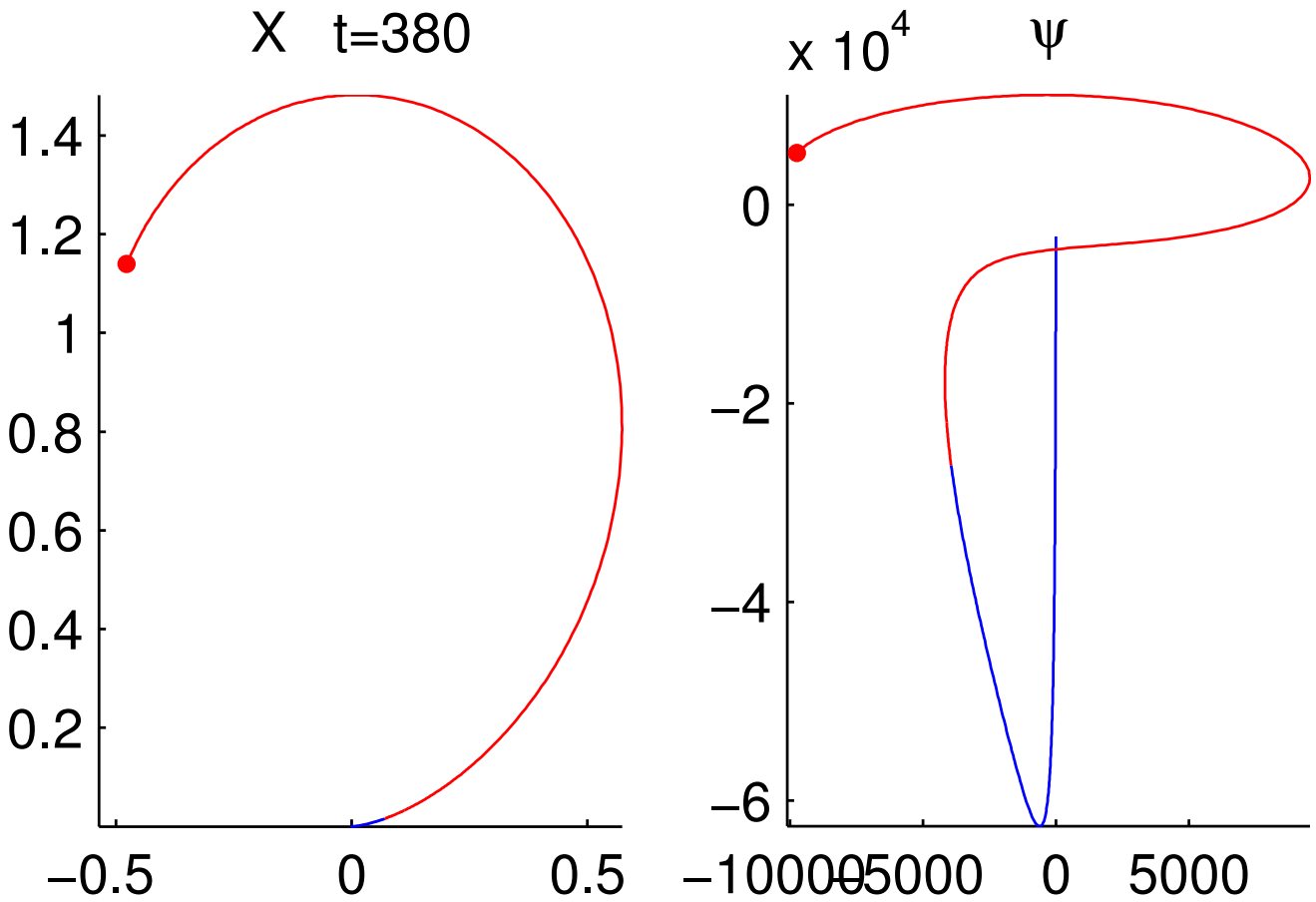
Transition to turbulence $\|X_0\| = 10^{-5.2}$



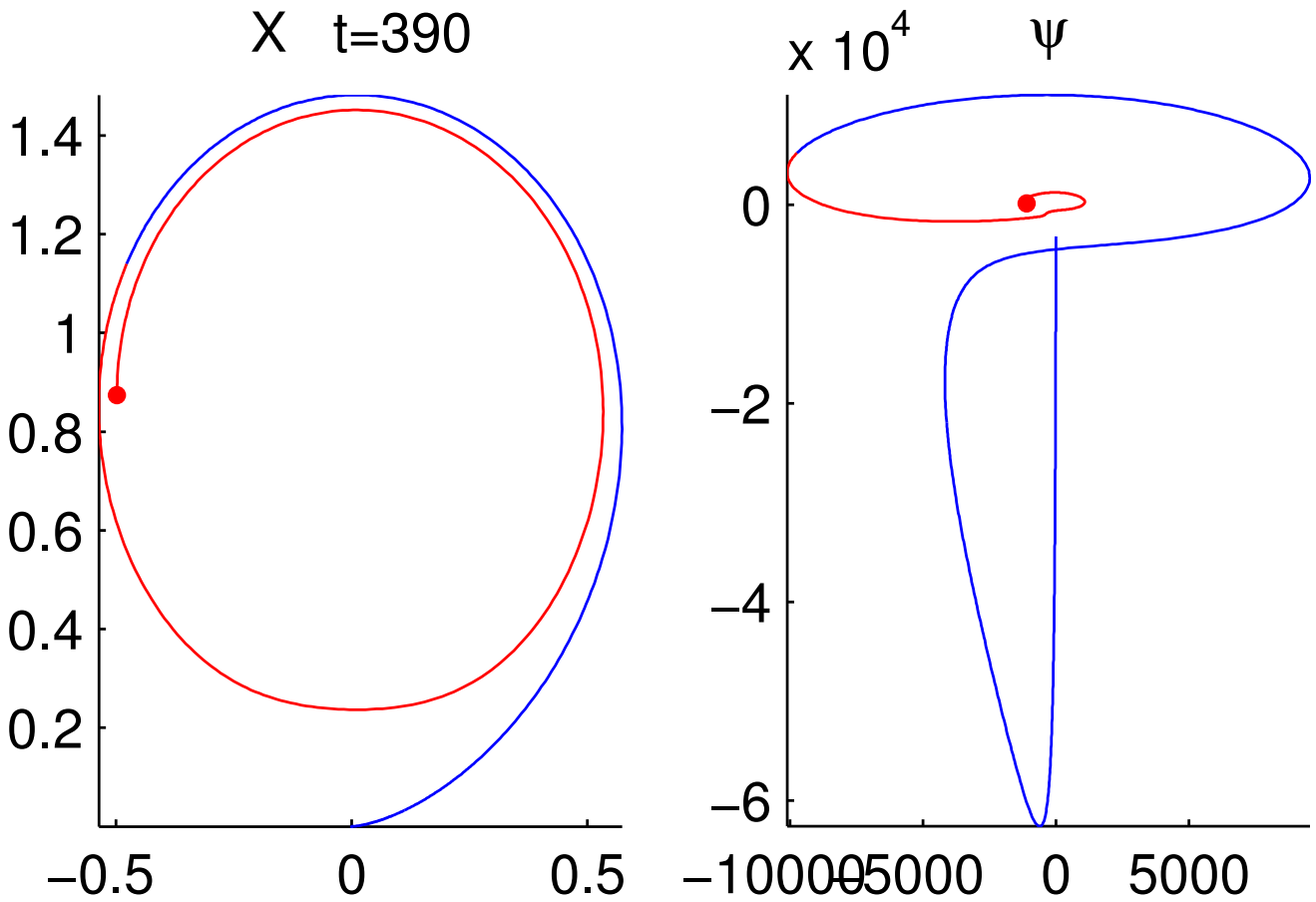
Transition to turbulence $\|X_0\| = 10^{-5.2}$



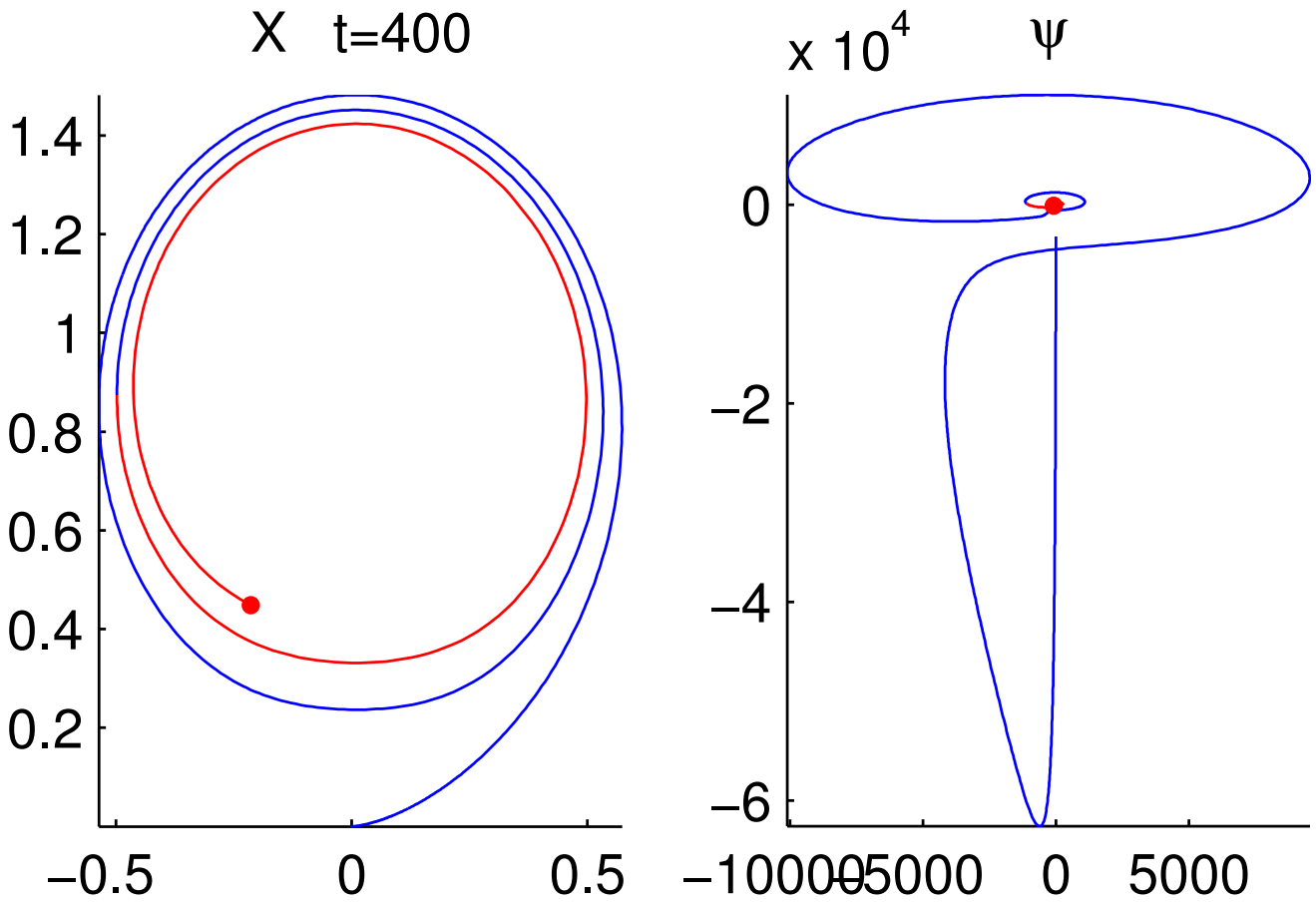
Transition to turbulence $\|X_0\| = 10^{-5.2}$



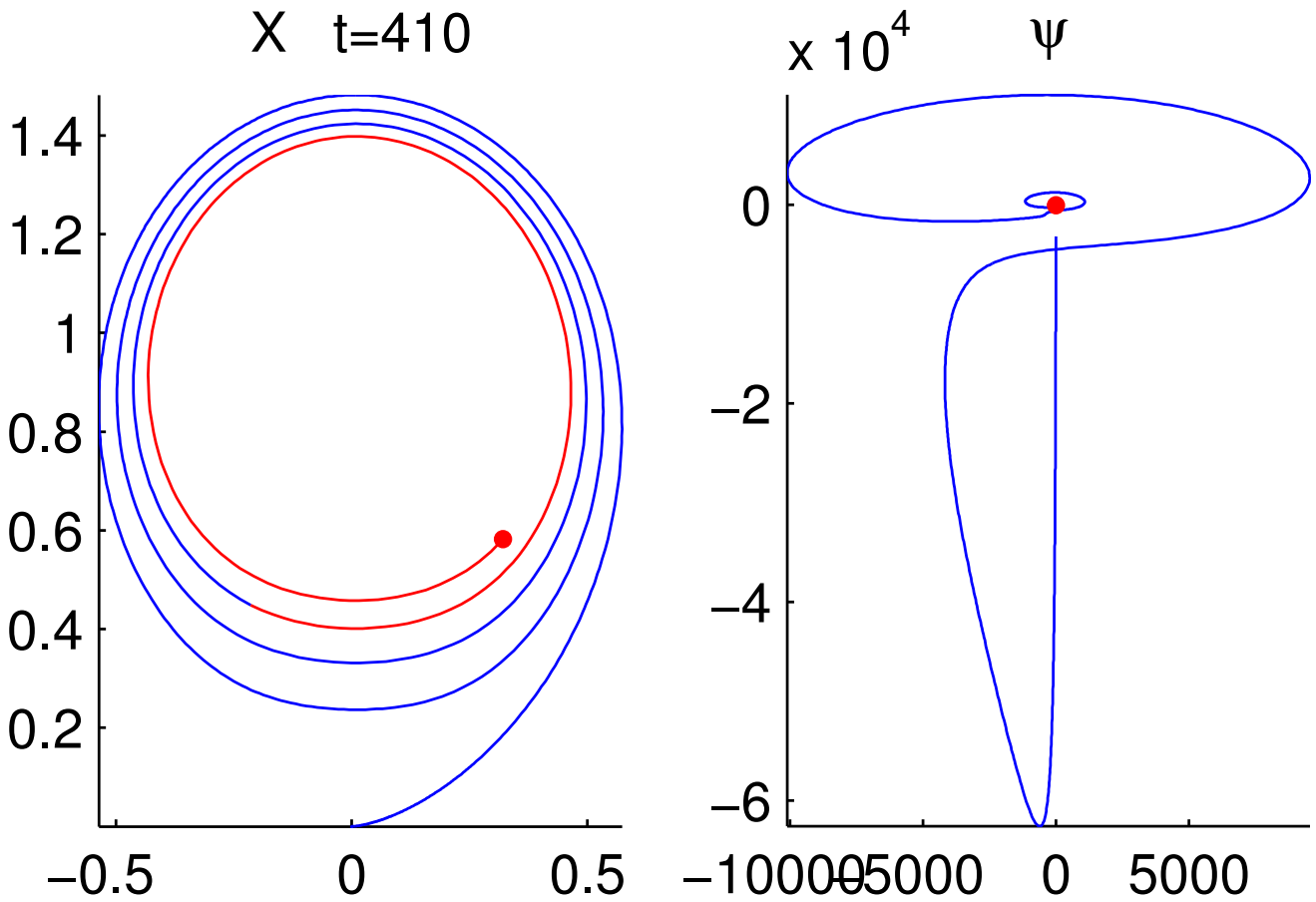
Transition to turbulence $\|X_0\| = 10^{-5.2}$



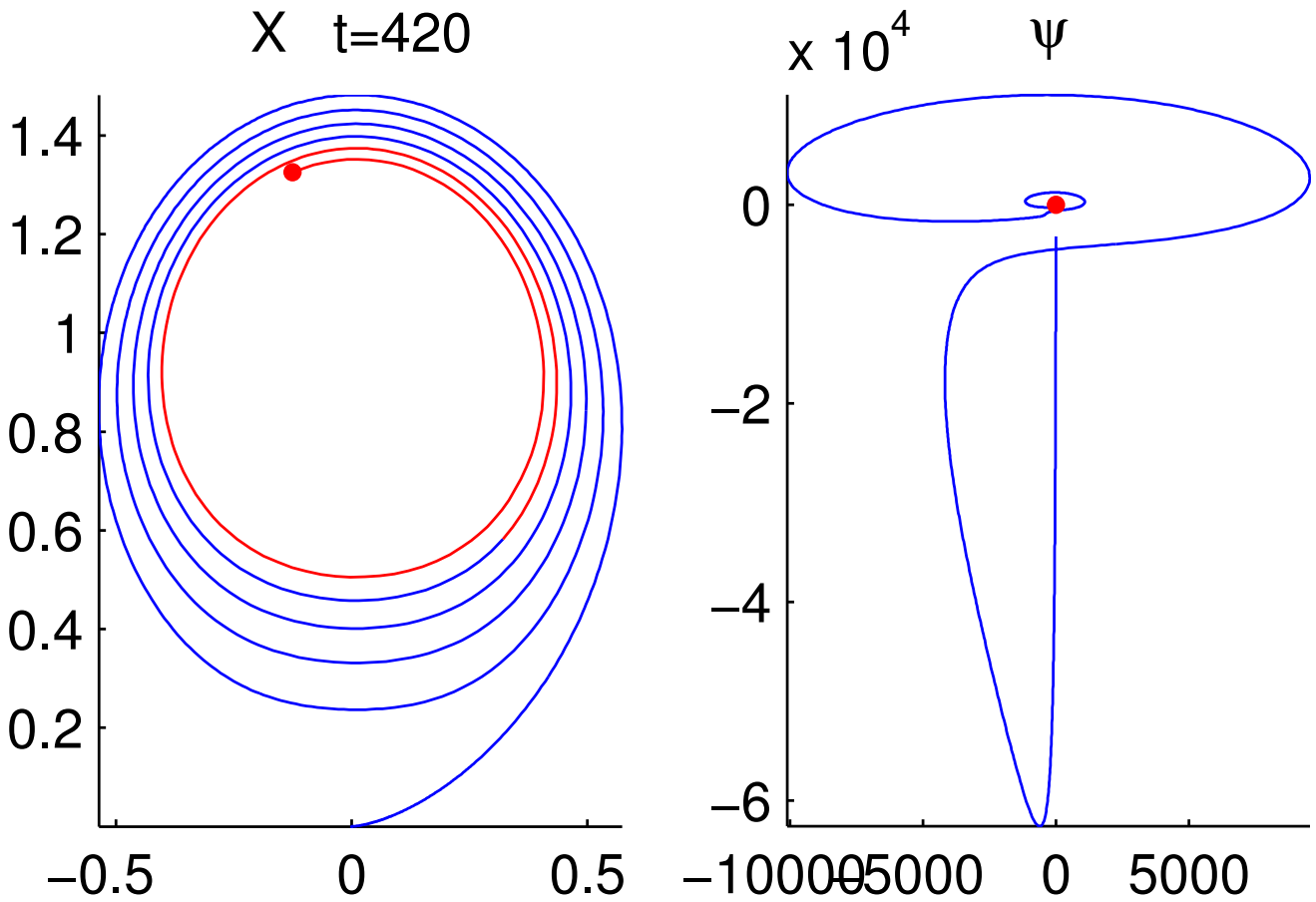
Transition to turbulence $\|X_0\| = 10^{-5.2}$



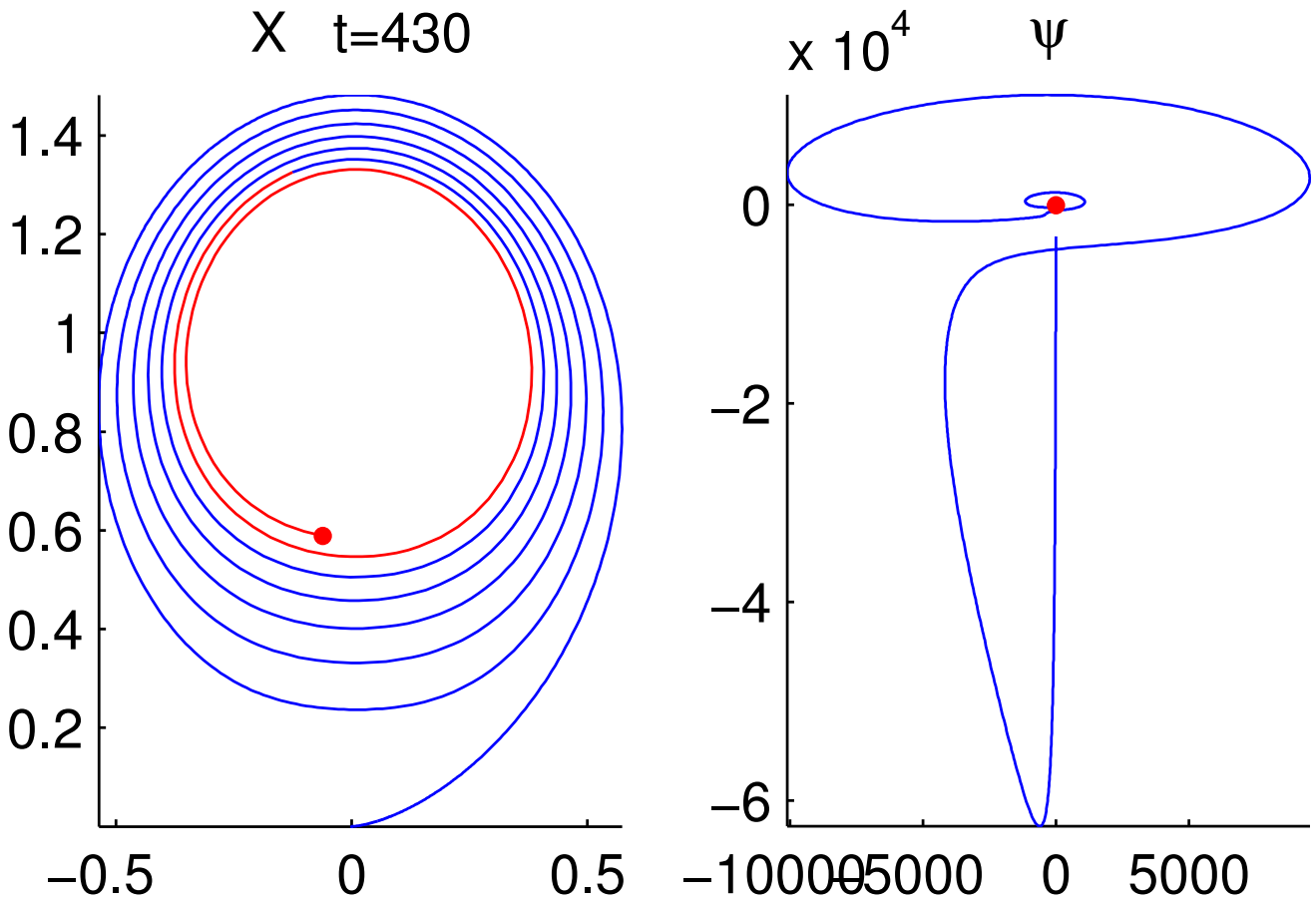
Transition to turbulence $\|X_0\| = 10^{-5.2}$



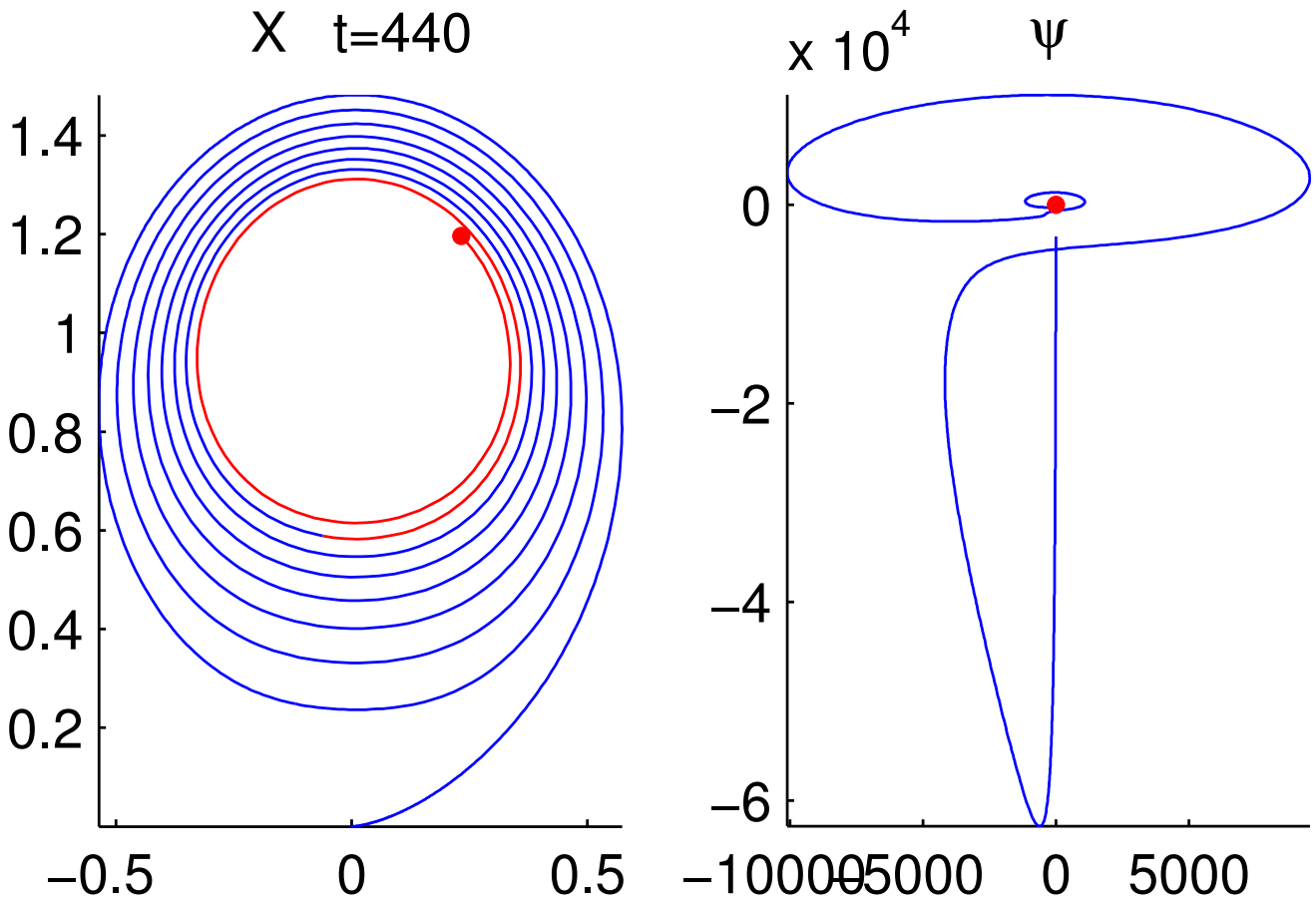
Transition to turbulence $\|X_0\| = 10^{-5.2}$



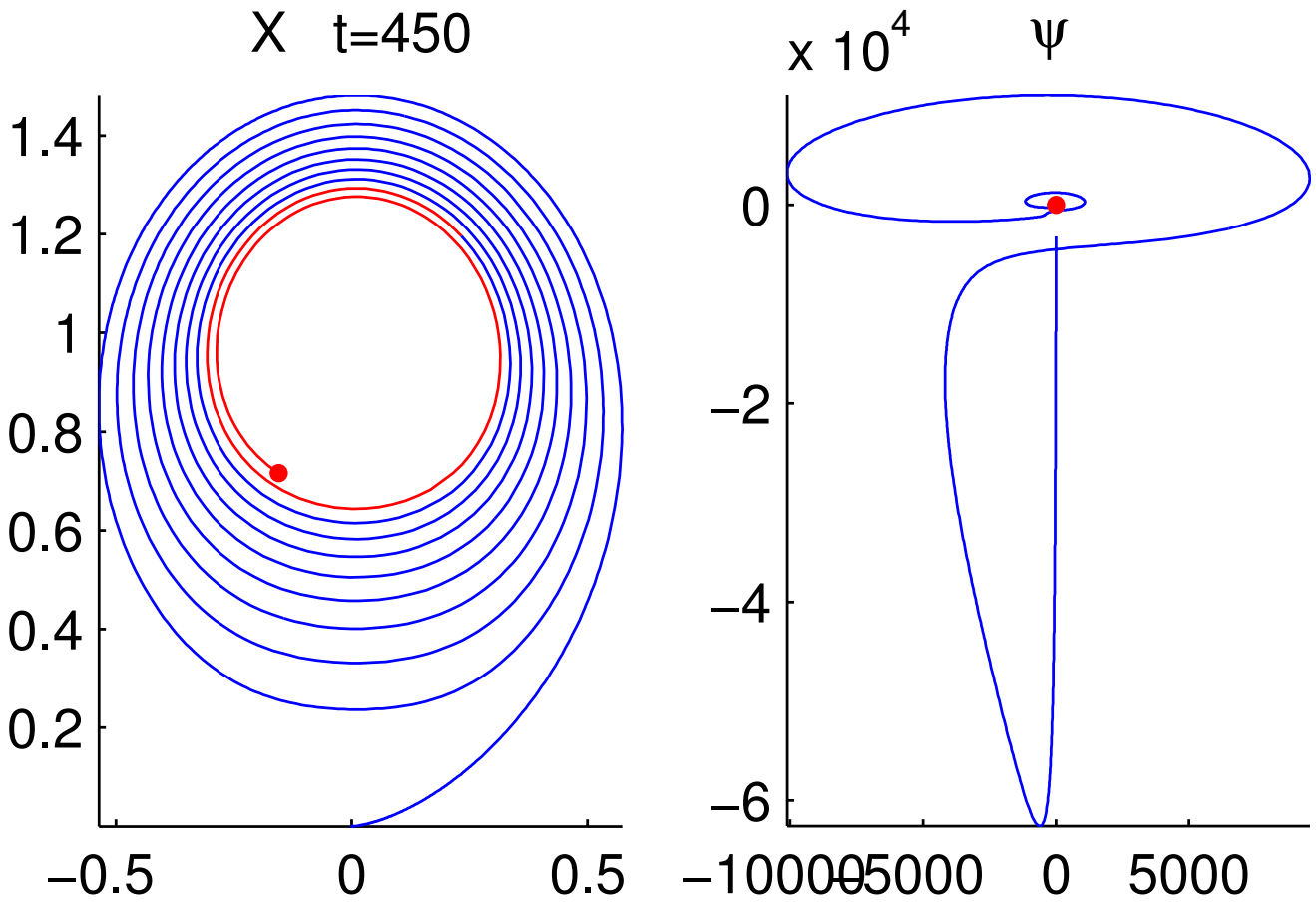
Transition to turbulence $\|X_0\| = 10^{-5.2}$



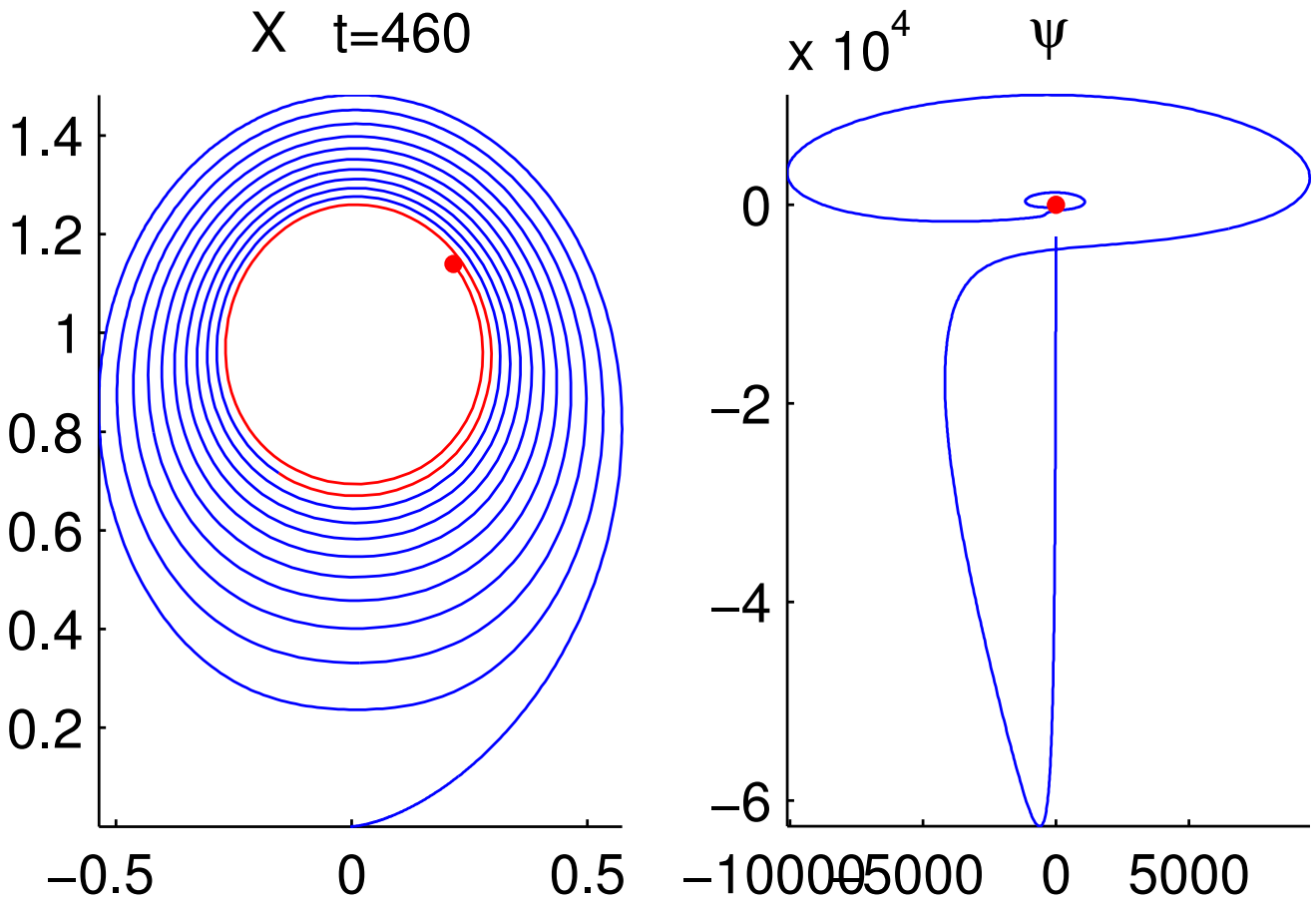
Transition to turbulence $\|X_0\| = 10^{-5.2}$



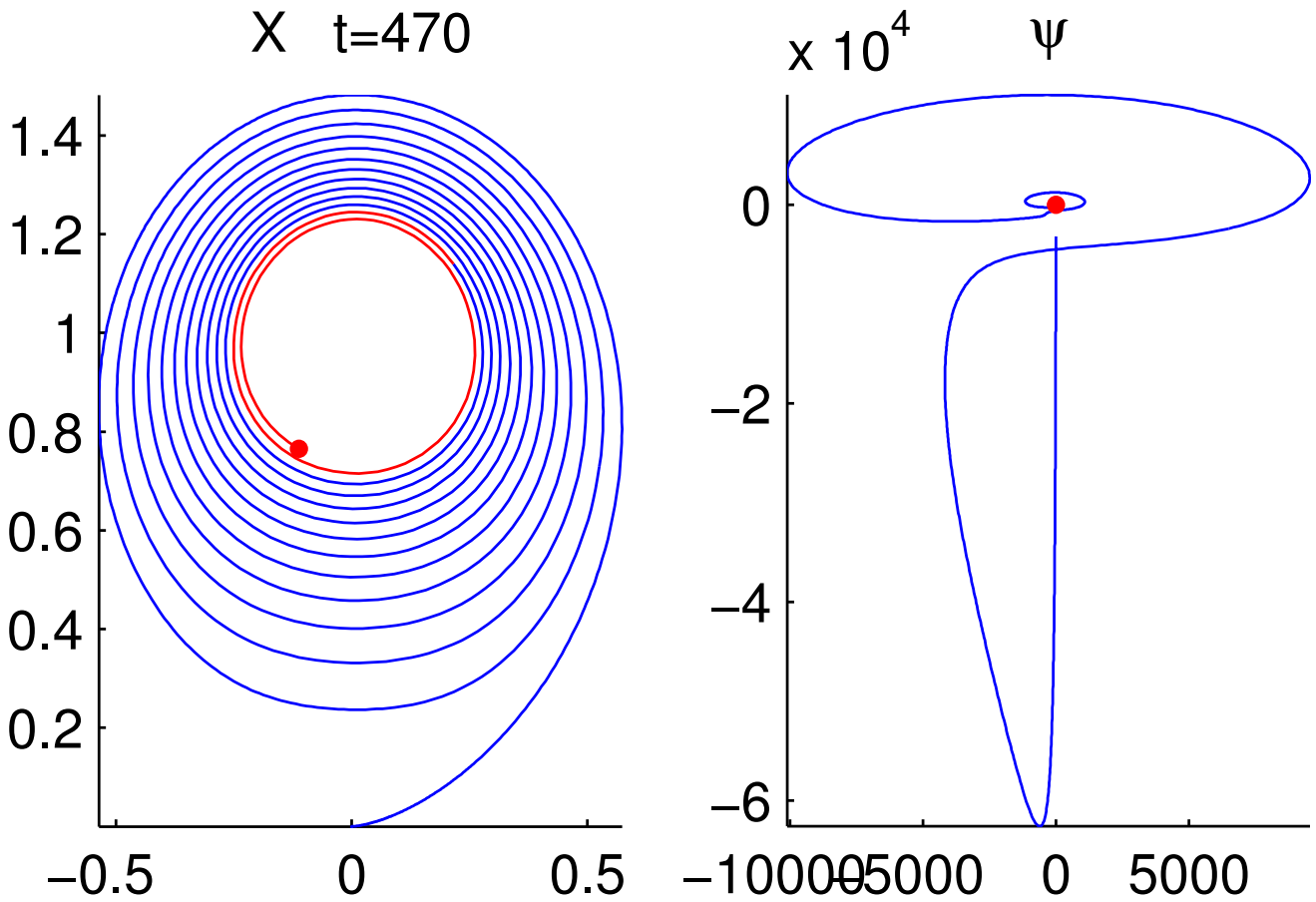
Transition to turbulence $\|X_0\| = 10^{-5.2}$



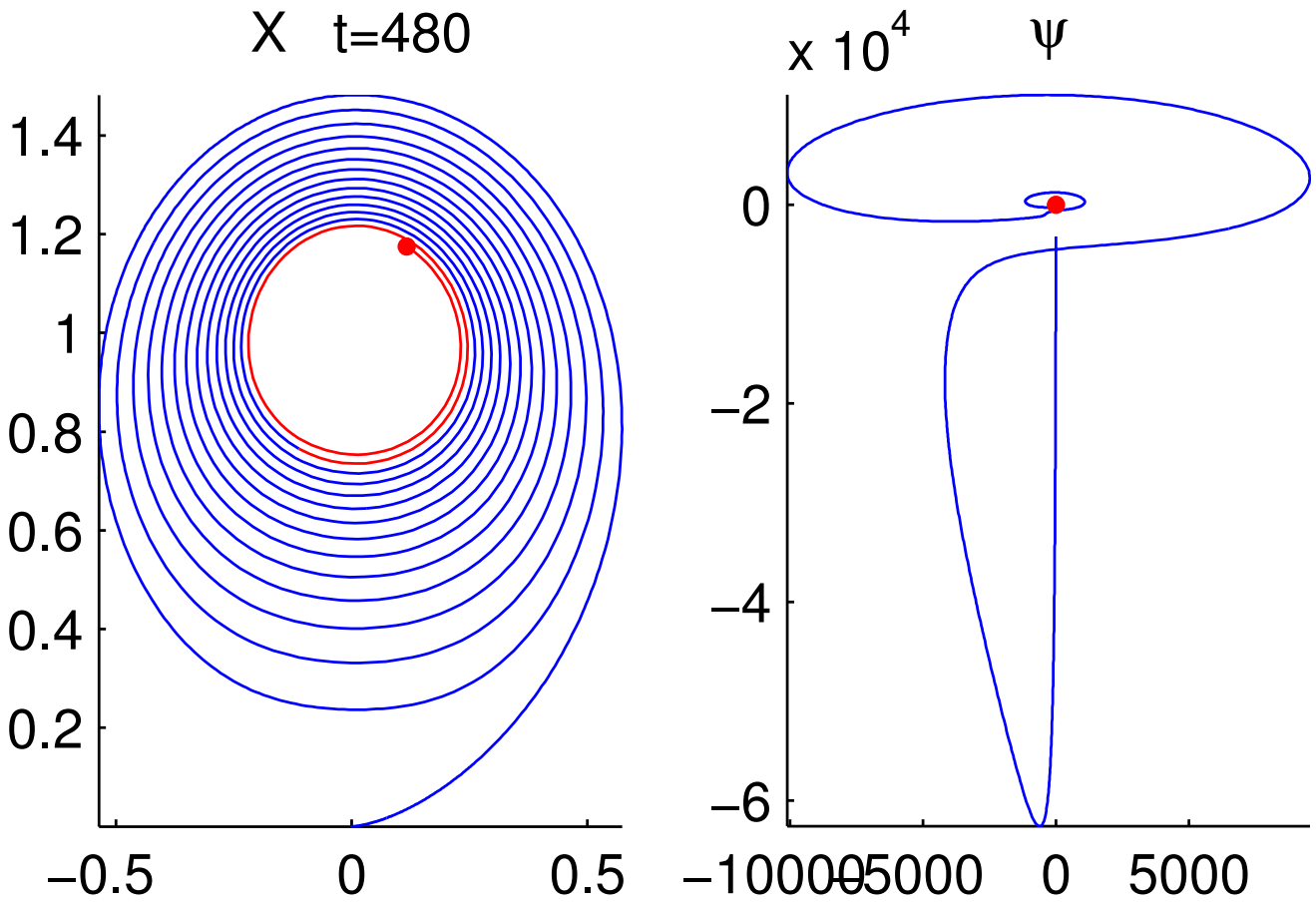
Transition to turbulence $\|X_0\| = 10^{-5.2}$



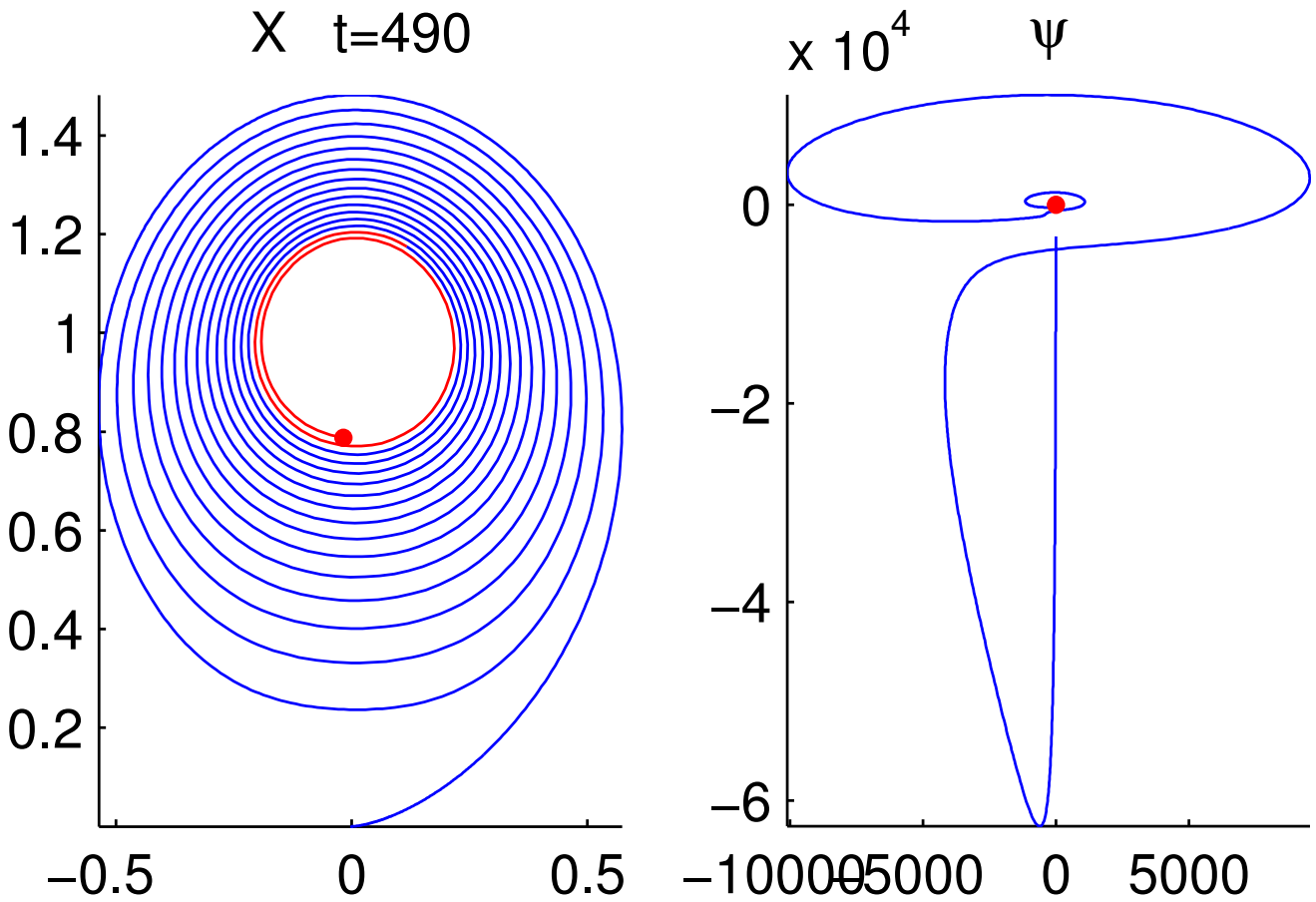
Transition to turbulence $\|X_0\| = 10^{-5.2}$



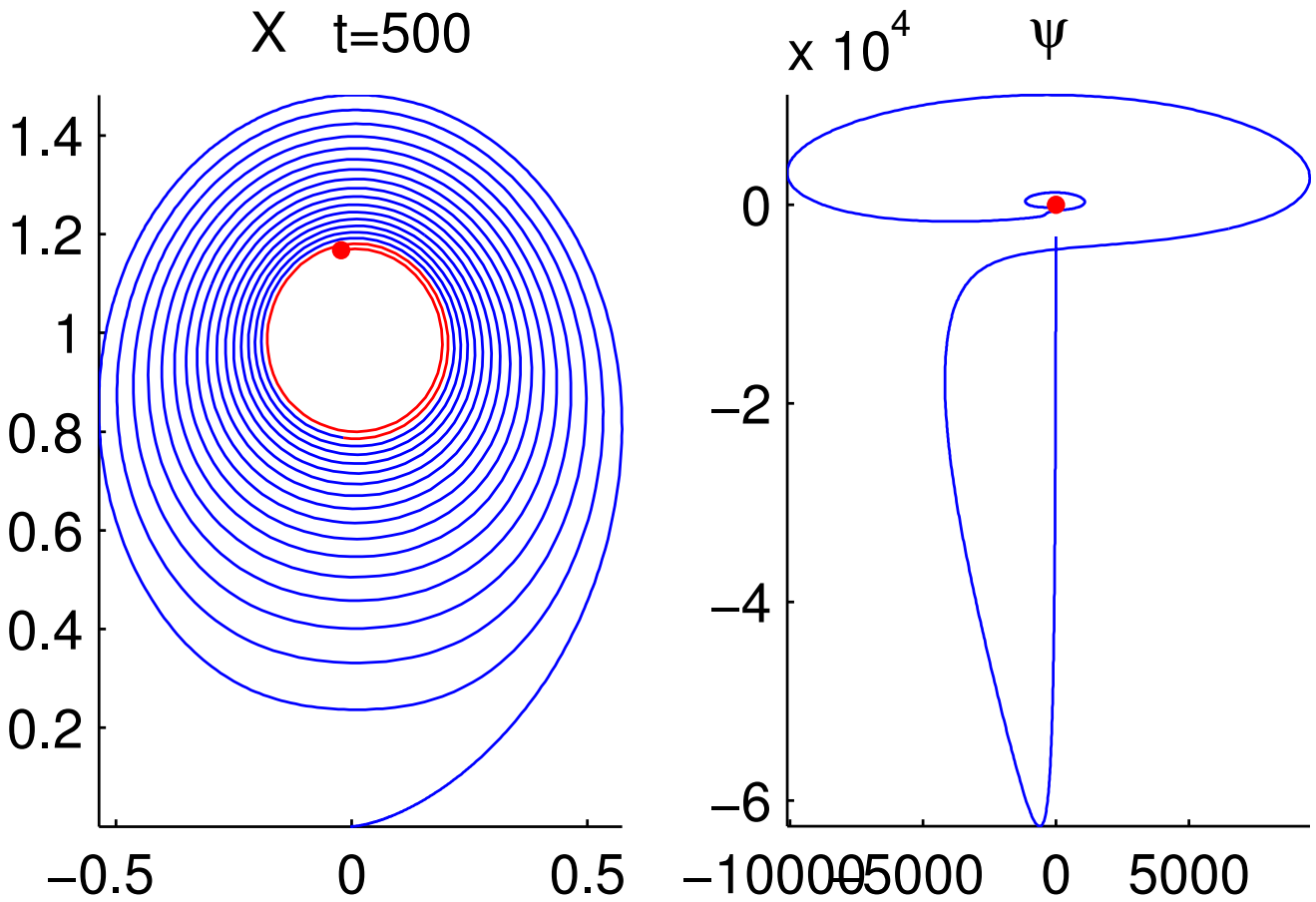
Transition to turbulence $\|X_0\| = 10^{-5.2}$



Transition to turbulence $\|X_0\| = 10^{-5.2}$

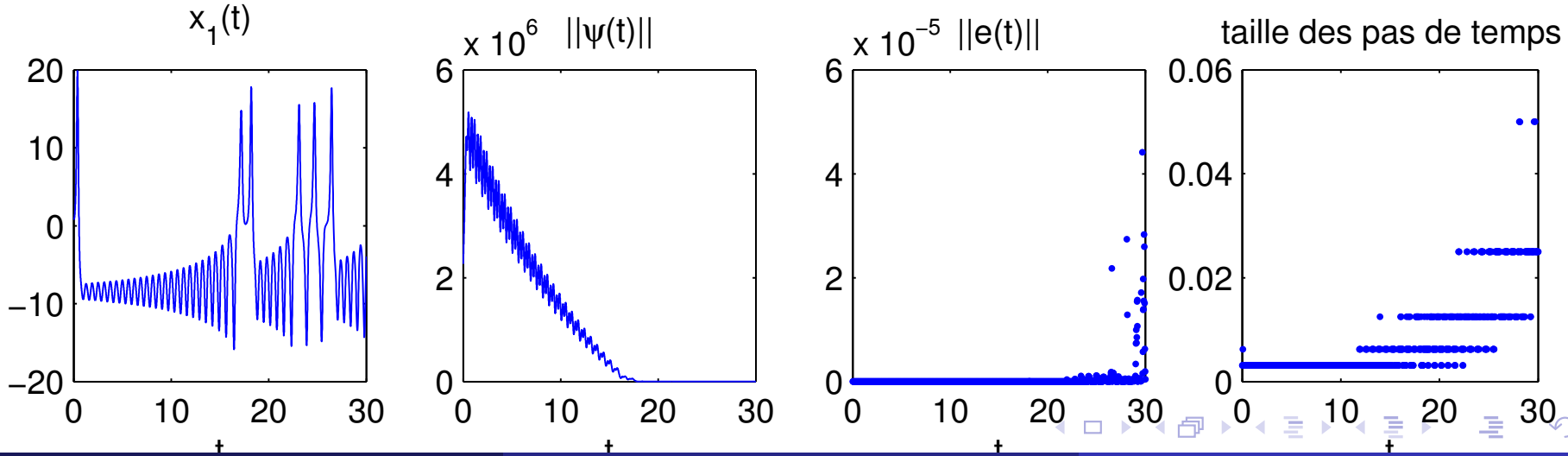


Transition to turbulence $\|X_0\| = 10^{-5.2}$

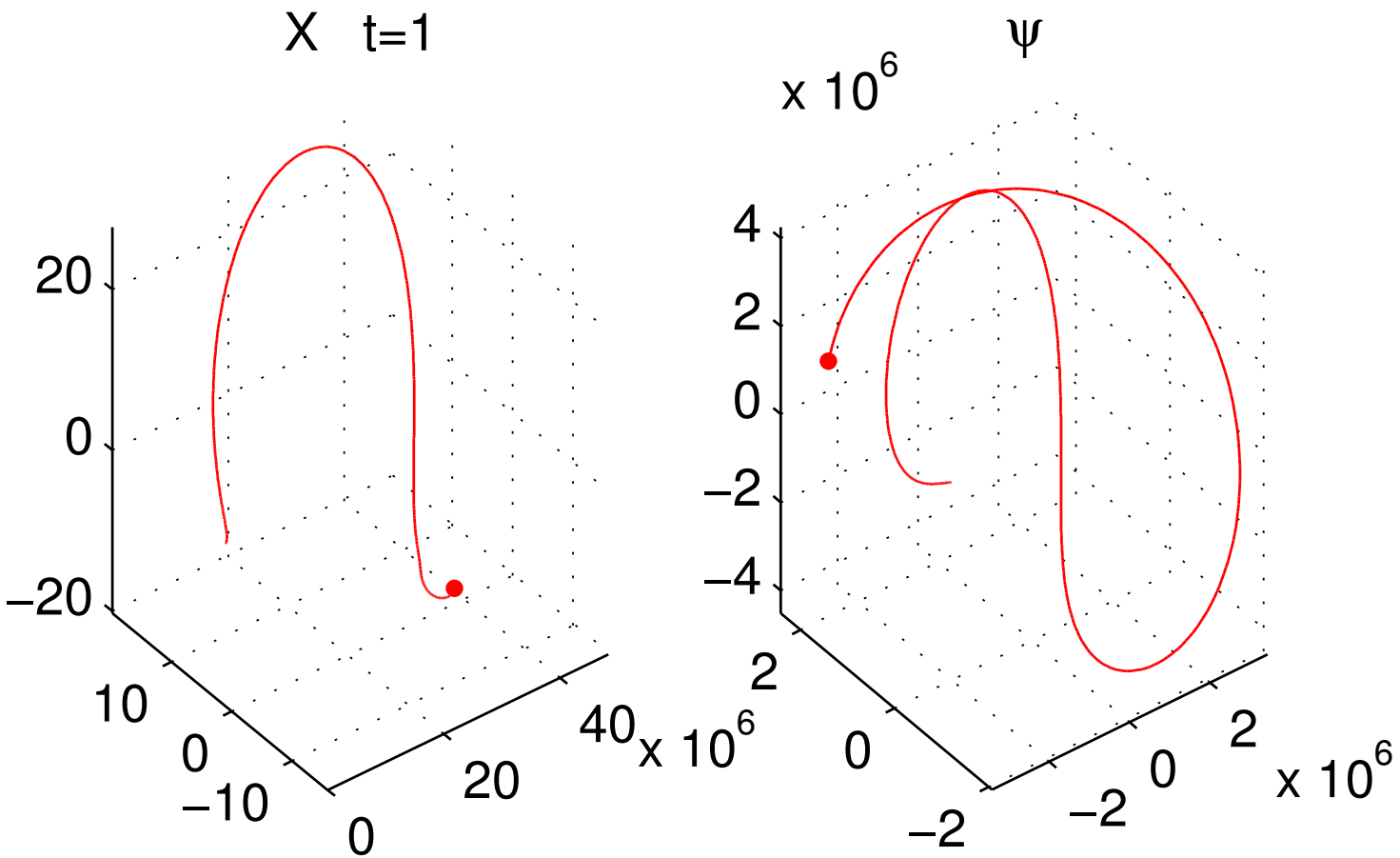


Lorenz Problem

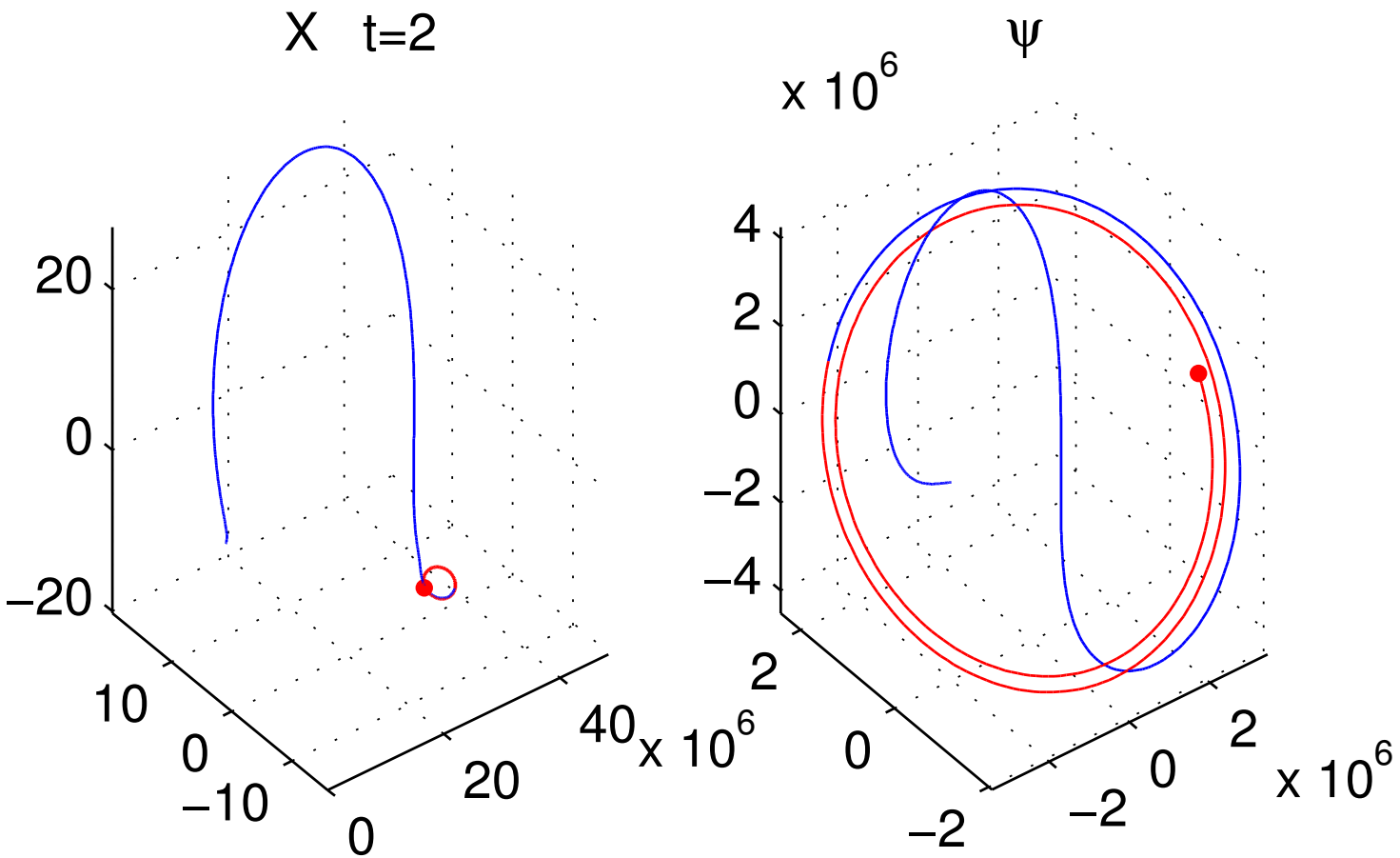
$$
(I)\begin{cases}
x_1'(t) &= -\sigma x_1(t) + \sigma x_2(t) \\
x_2'(t) &= r x_1(t) - x_2(t) - x_1(t)x_3(t) \\
x_3'(t) &= x_1(t)x_2(t) - b x_3(t) \\
 & \forall t \in [0\ 10] \text{ avec } \sigma = 10,\ b = 8/3,\ r = 28 \\
X(0) &= (1,0,0) \\
g(X) &= x_1 \\
\text{TOL} &= 0.1 \text{ et } 0.01 \\
N_0 &= 300
\end{cases}
$$



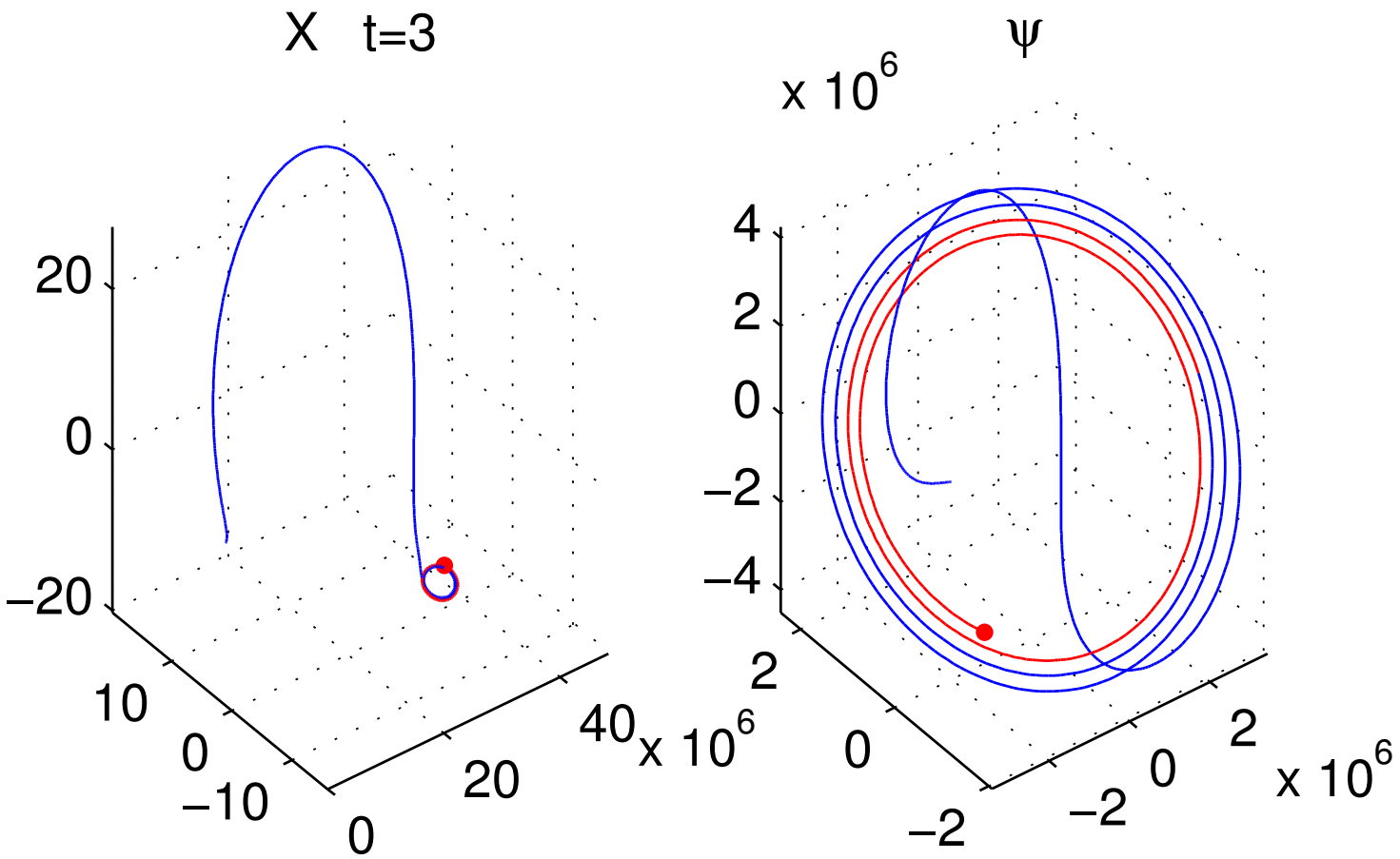
Lorenz with Dual



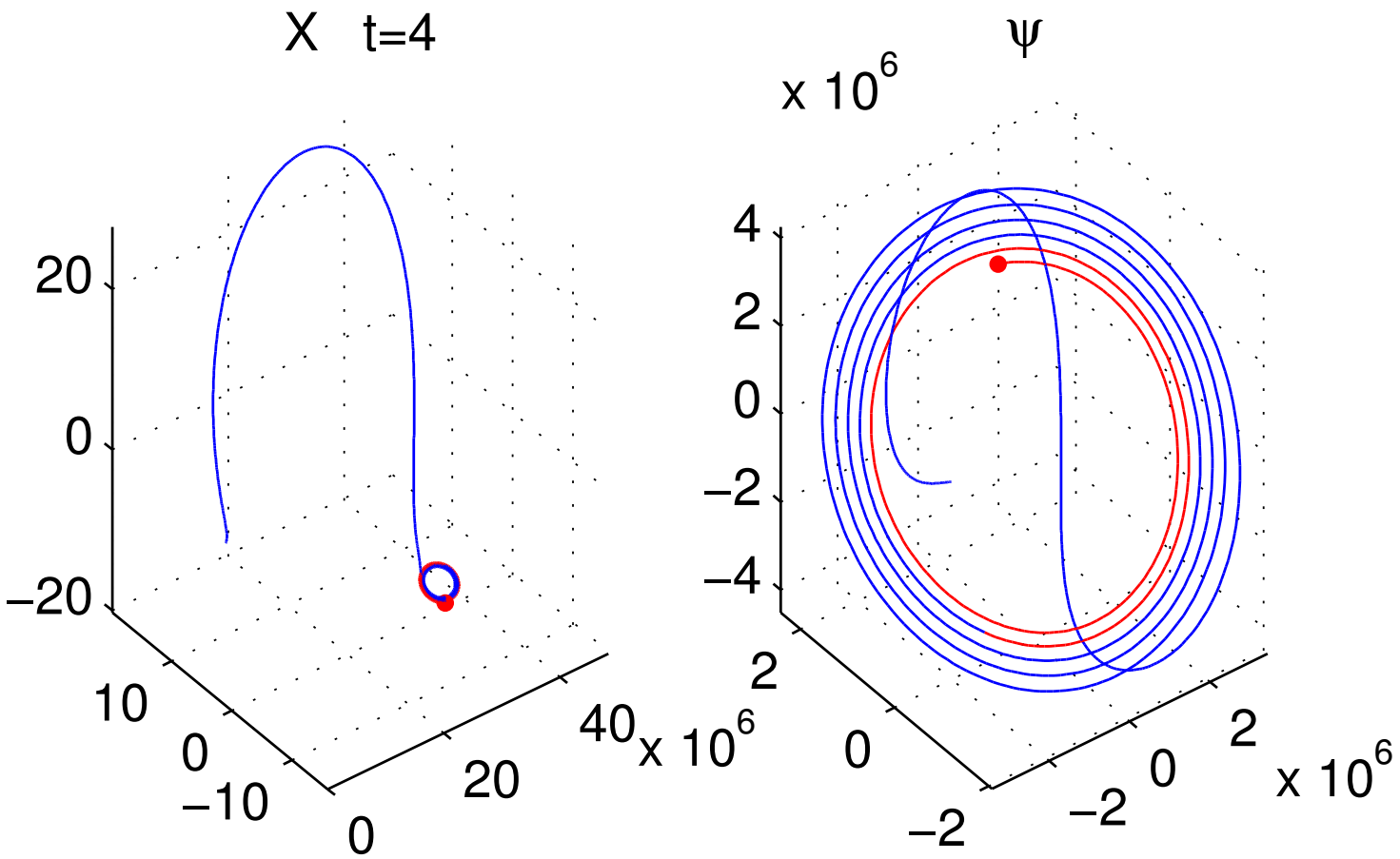
Lorenz with Dual



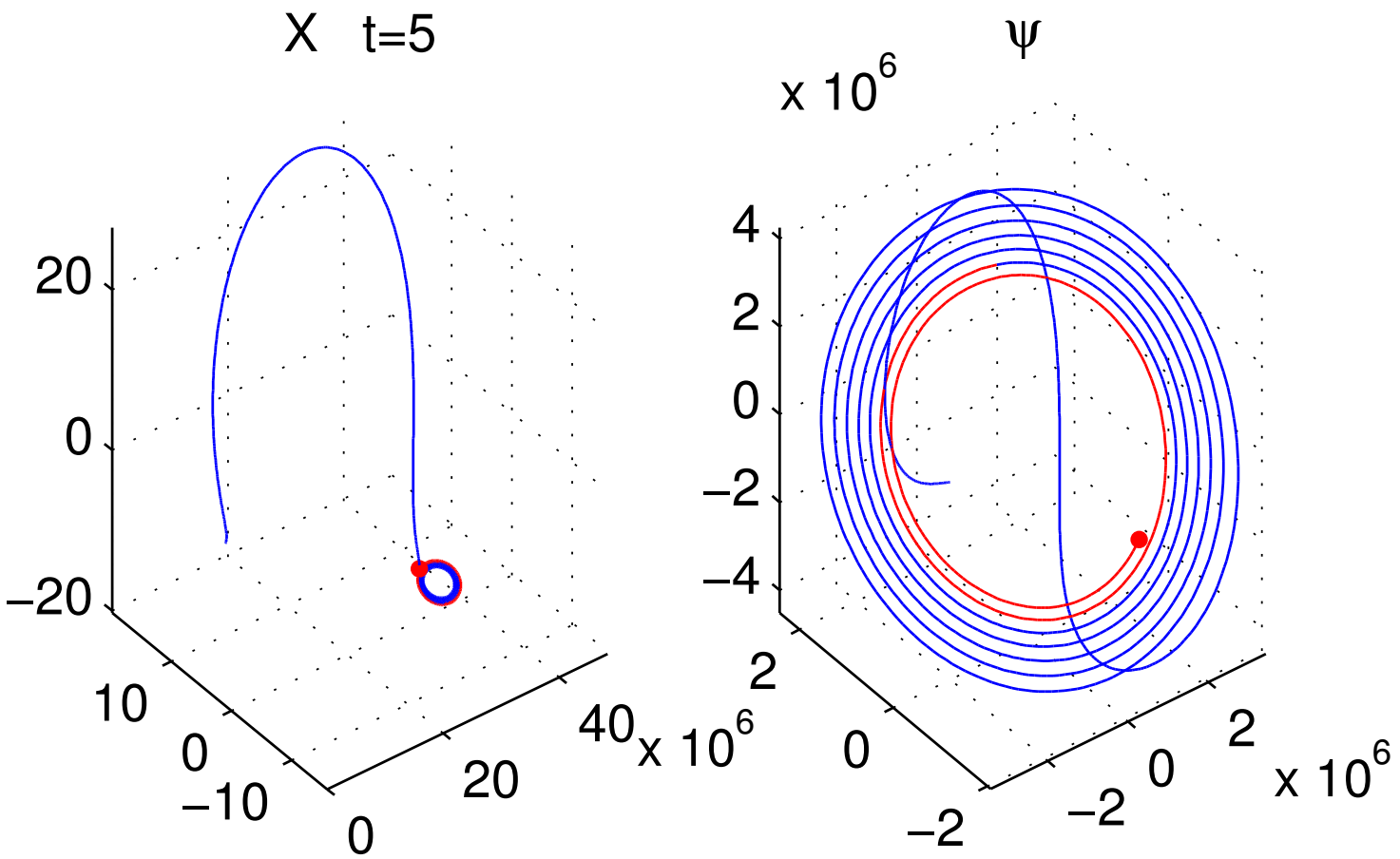
Lorenz with Dual



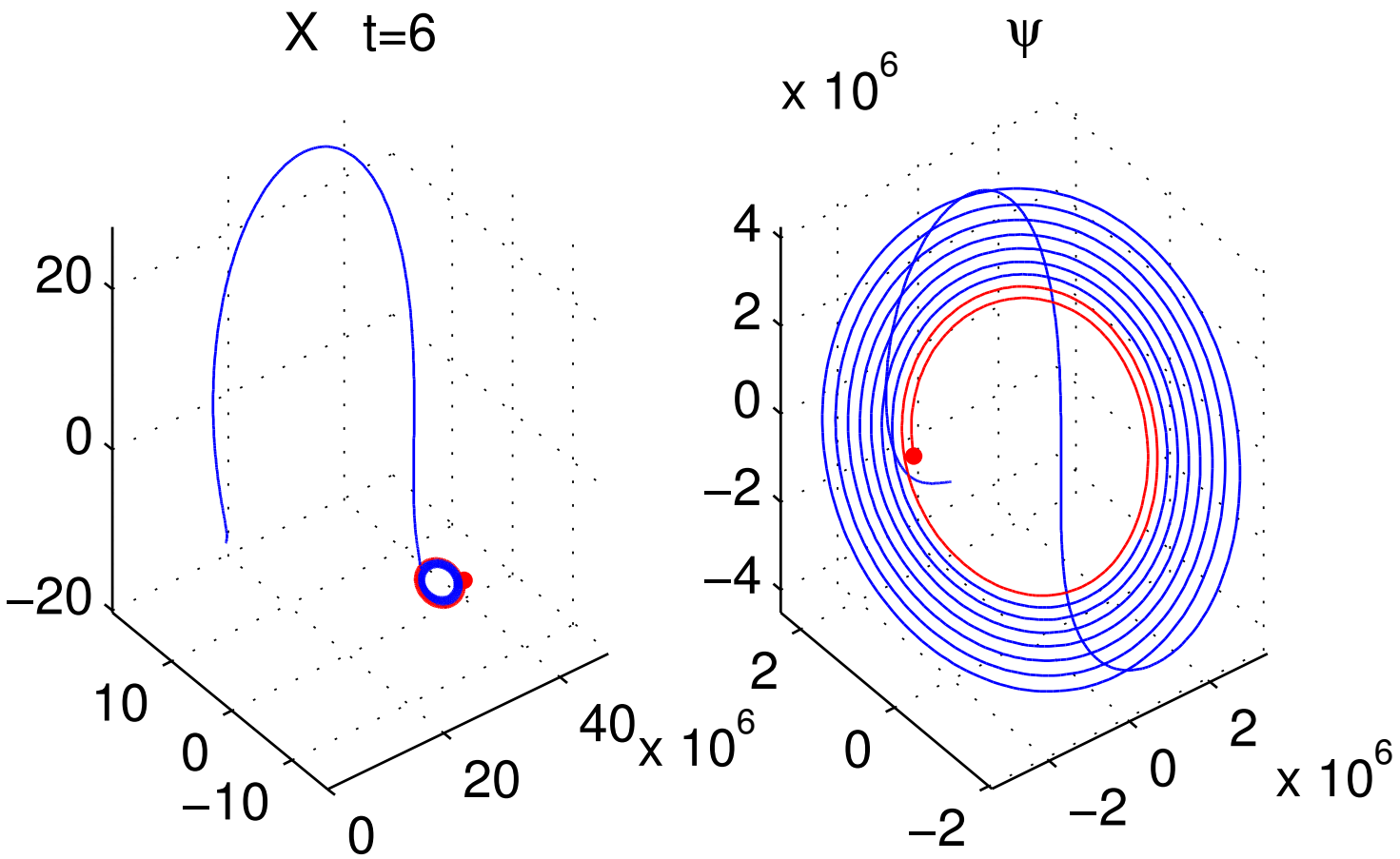
Lorenz with Dual



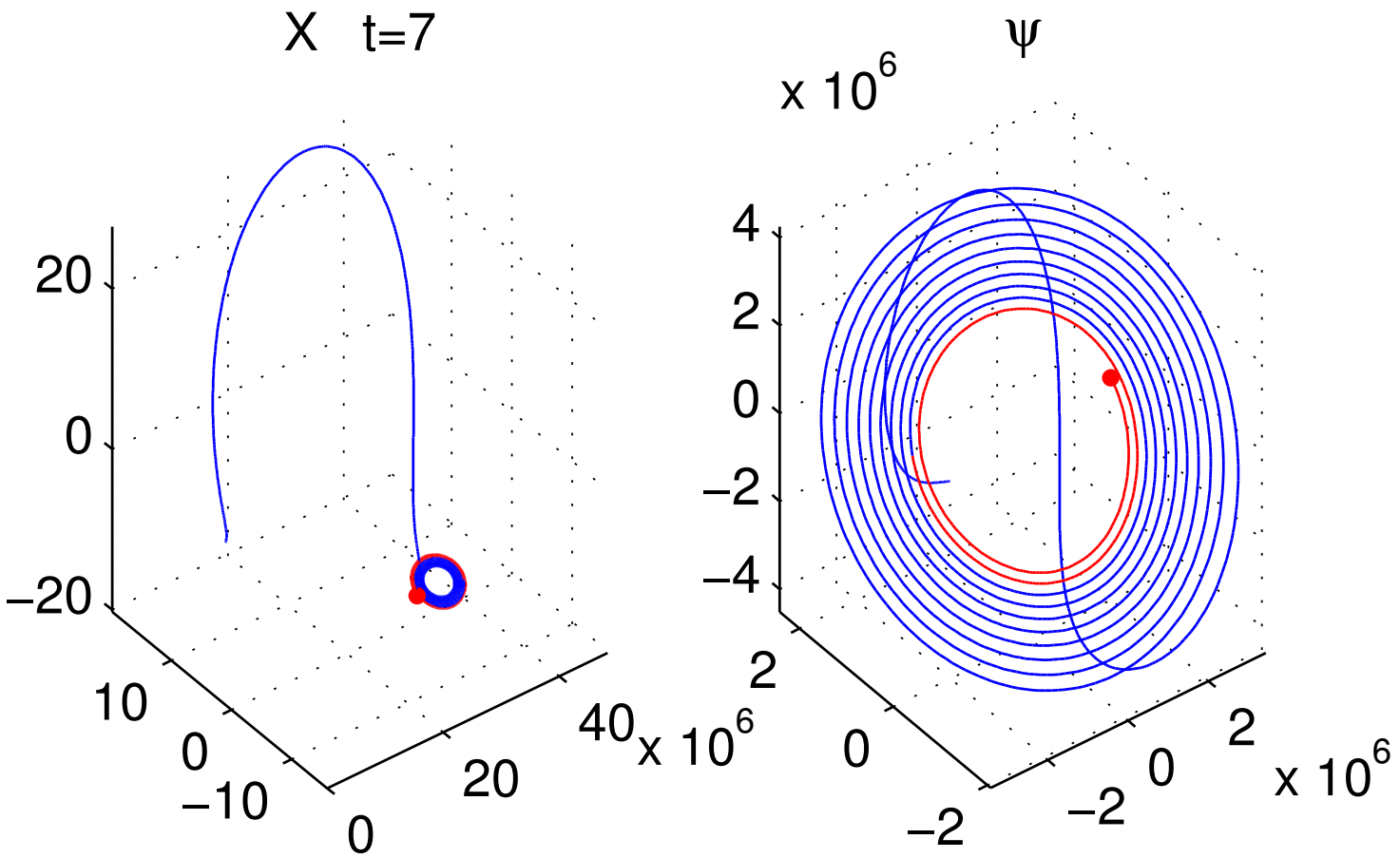
Lorenz with Dual



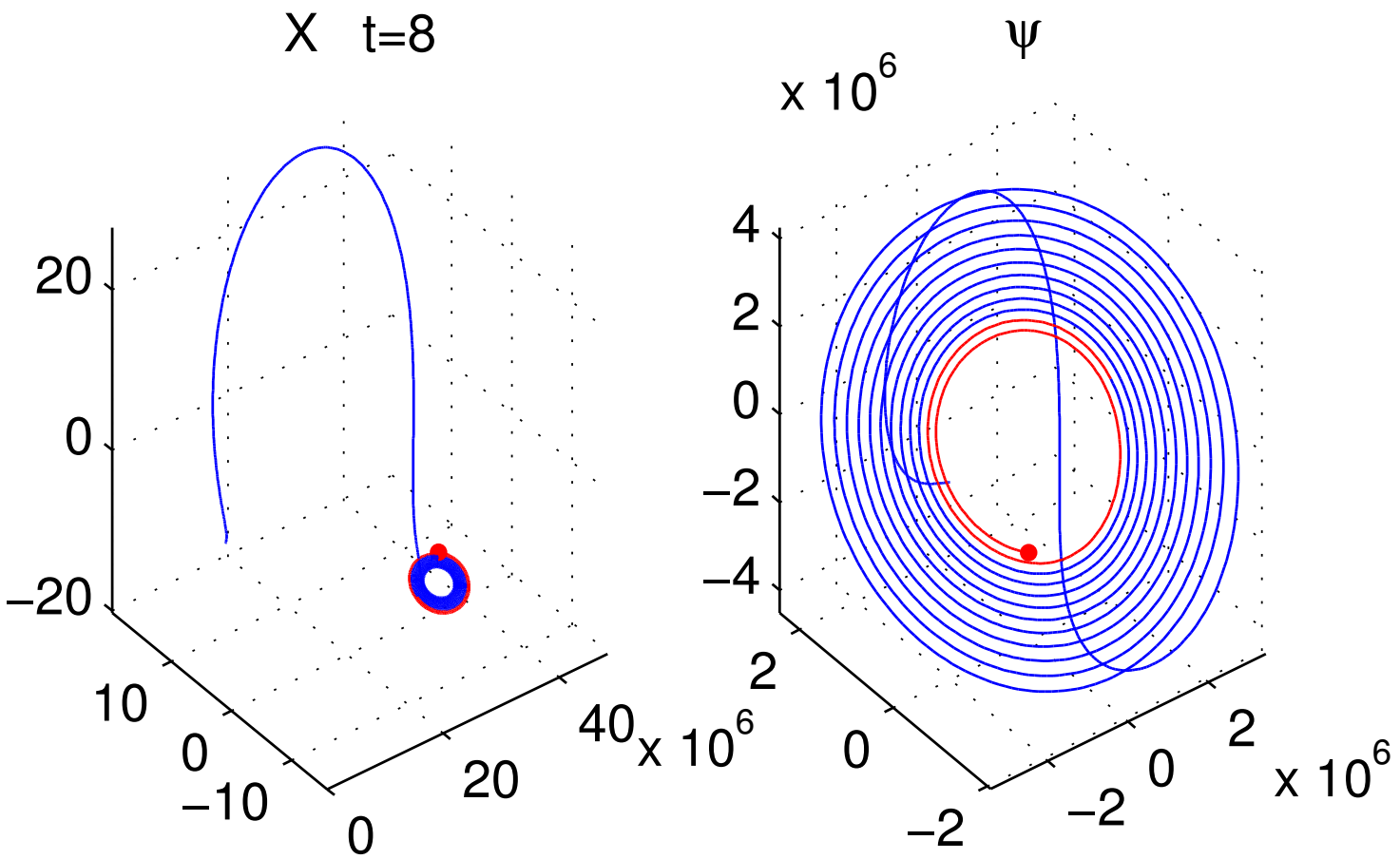
Lorenz with Dual



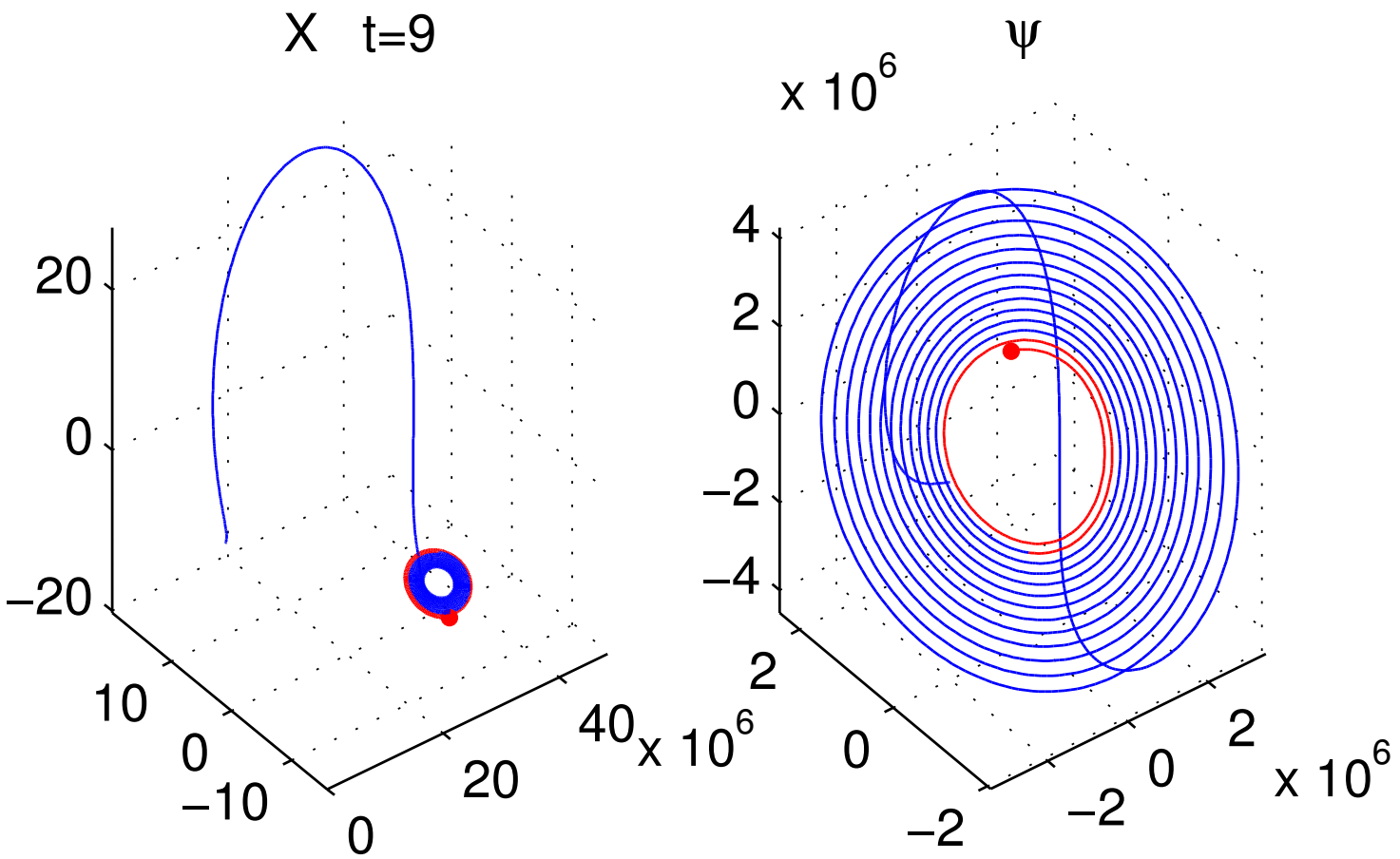
Lorenz with Dual



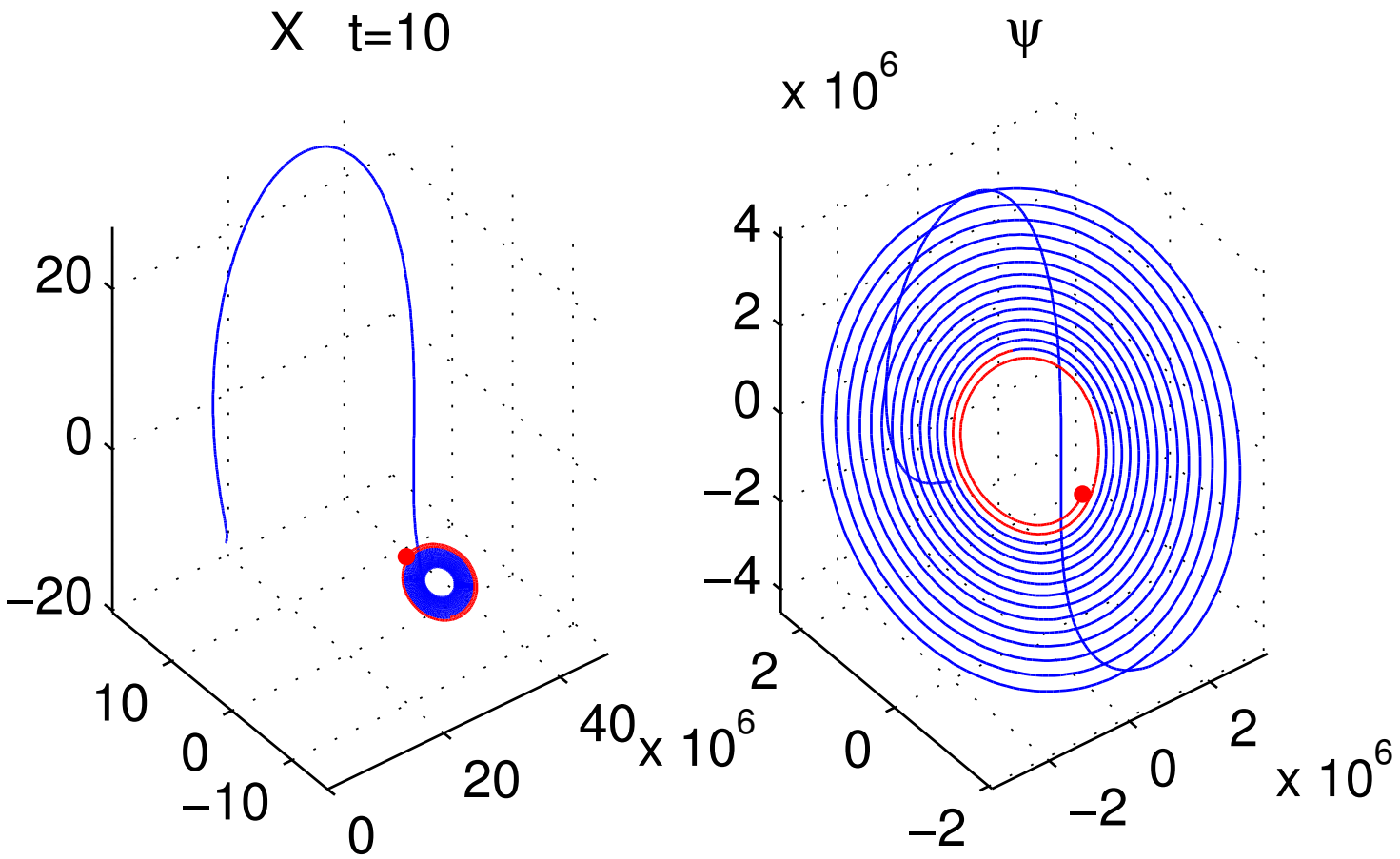
Lorenz with Dual



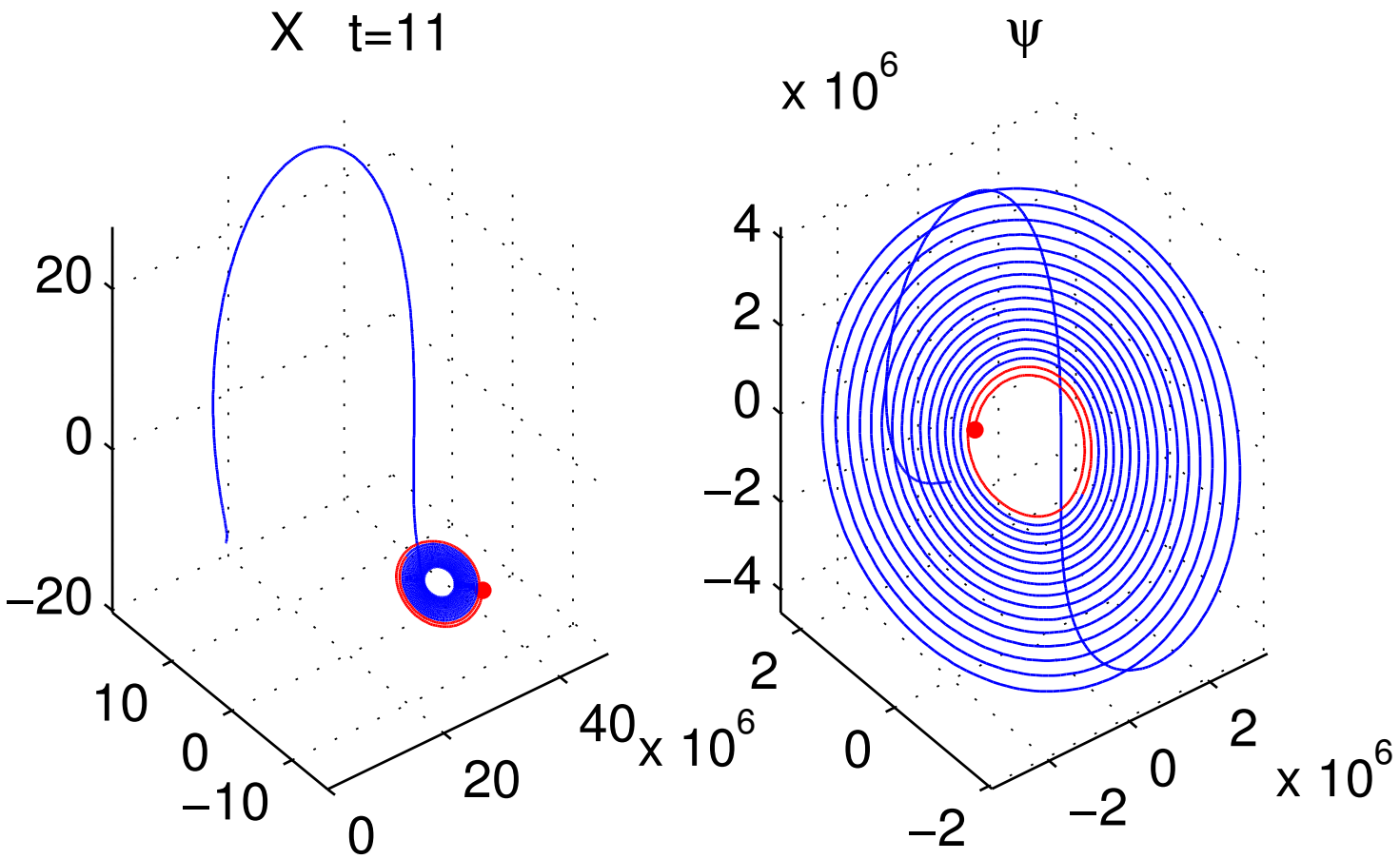
Lorenz with Dual



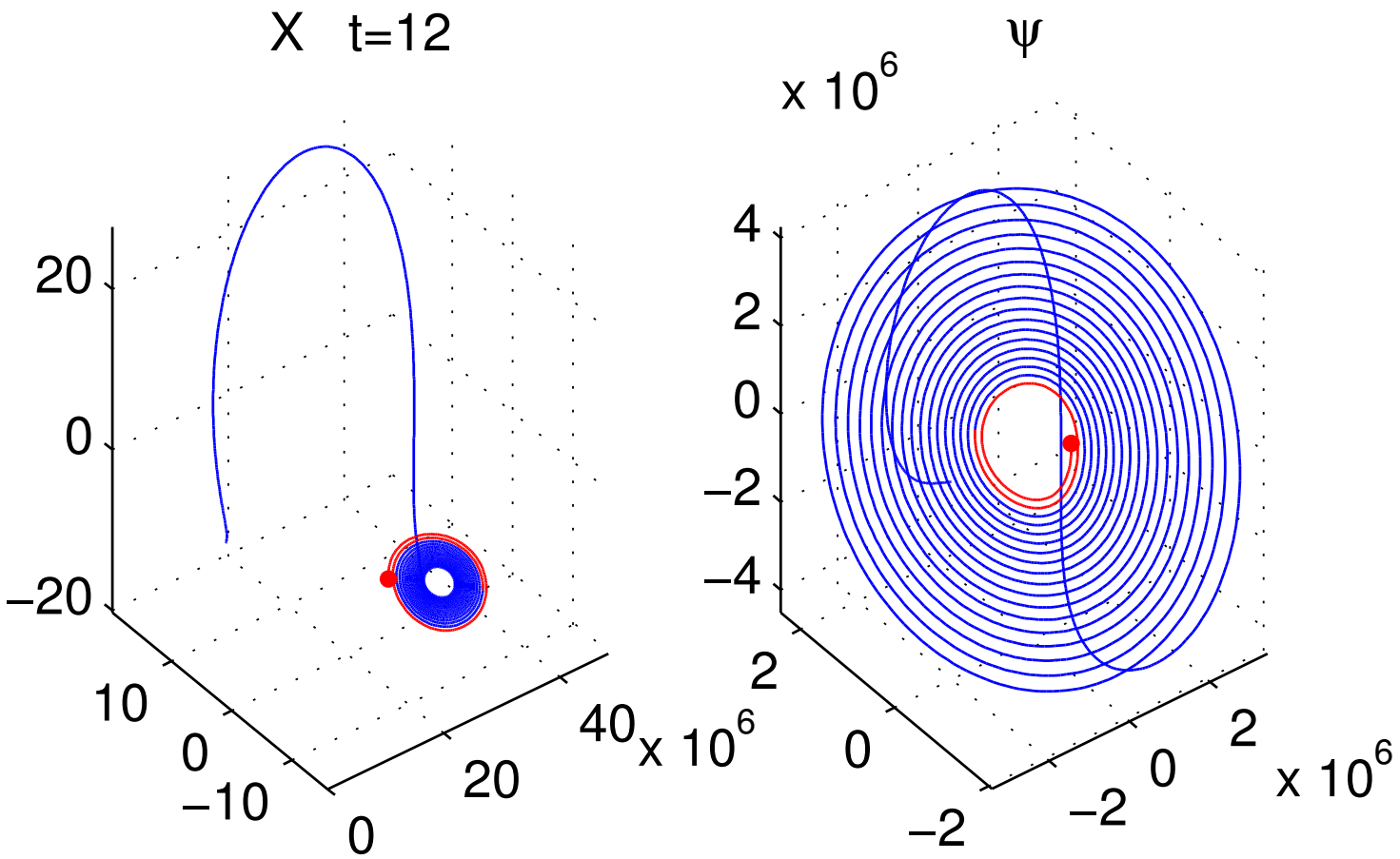
Lorenz with Dual



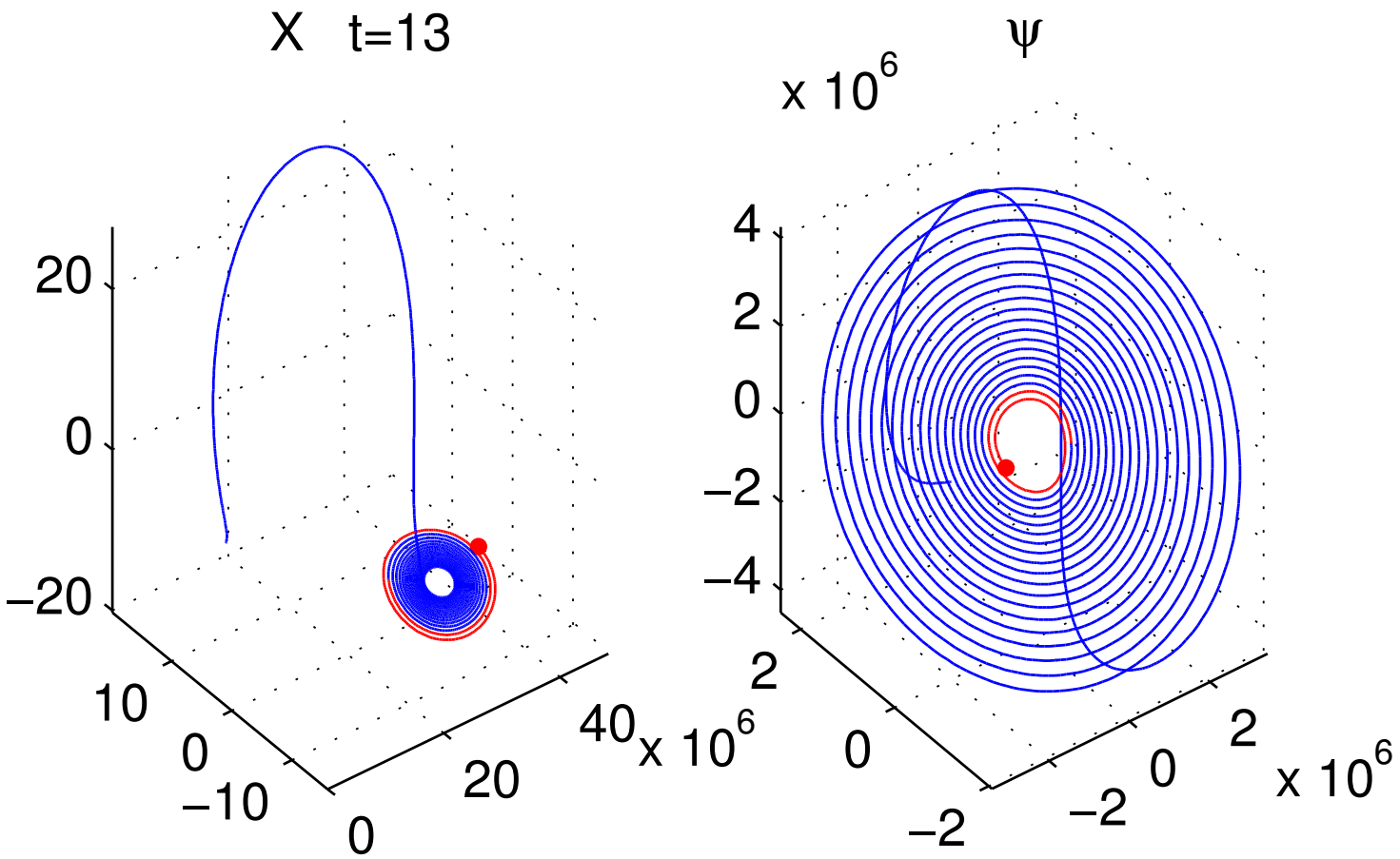
Lorenz with Dual



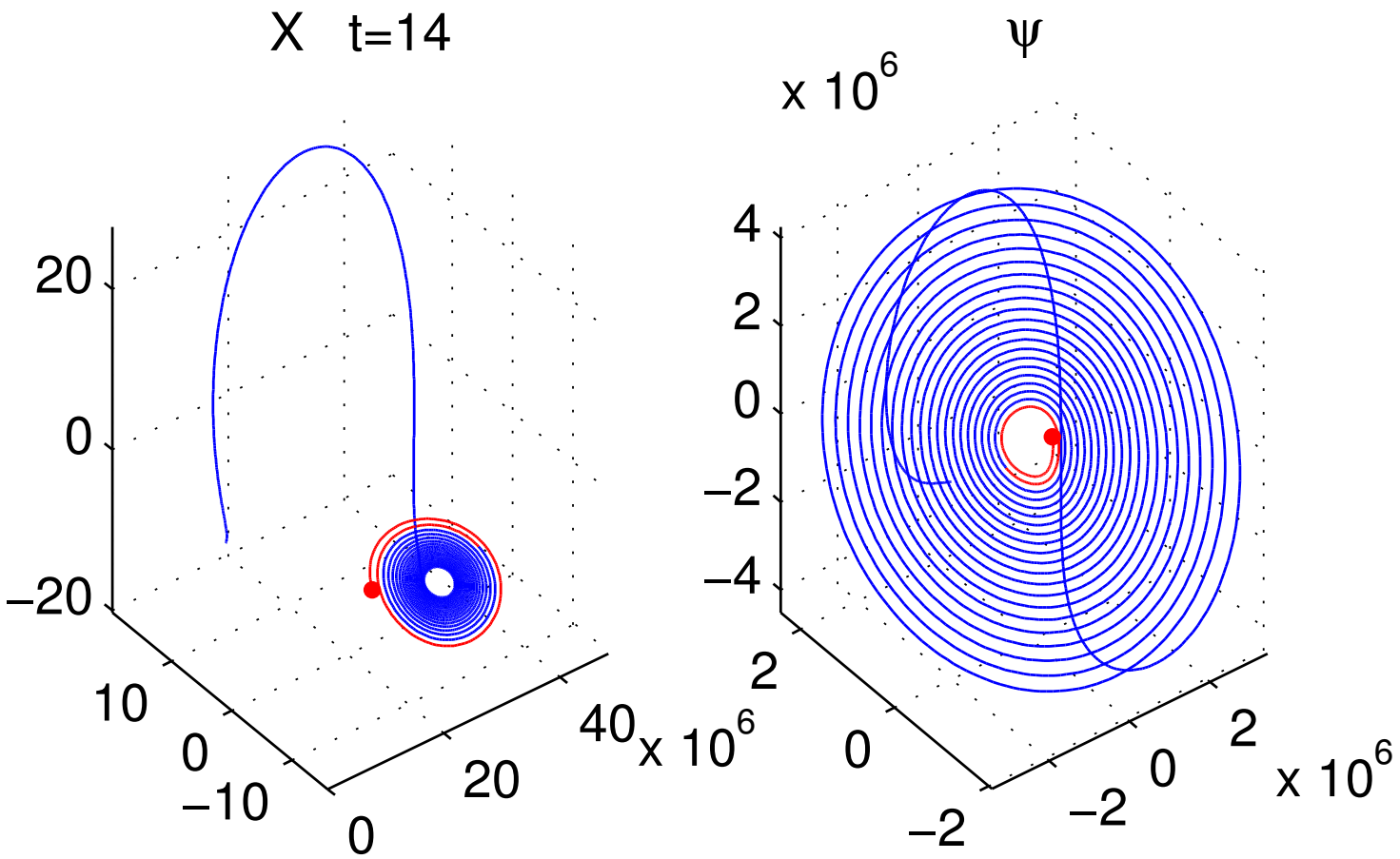
Lorenz with Dual



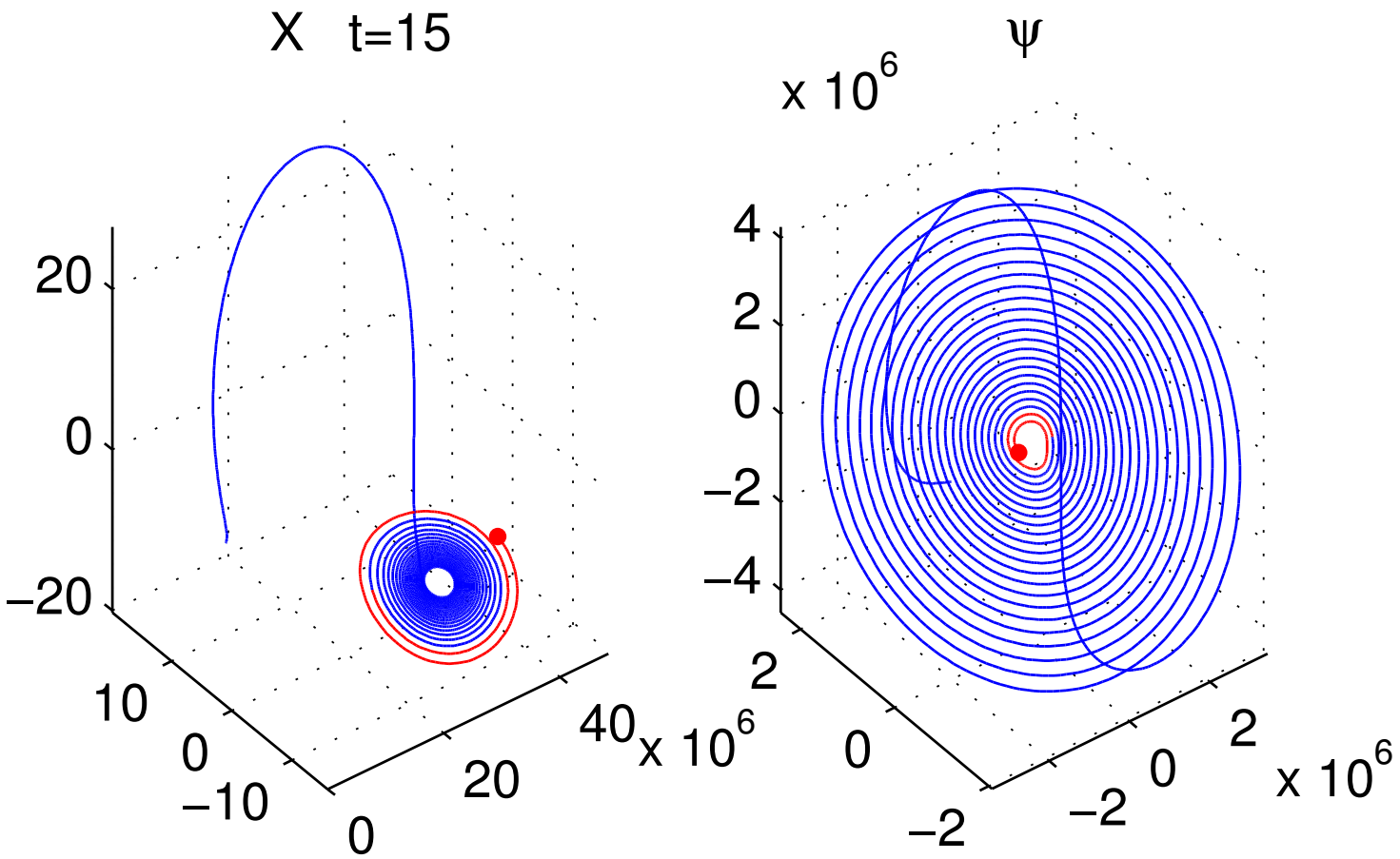
Lorenz with Dual



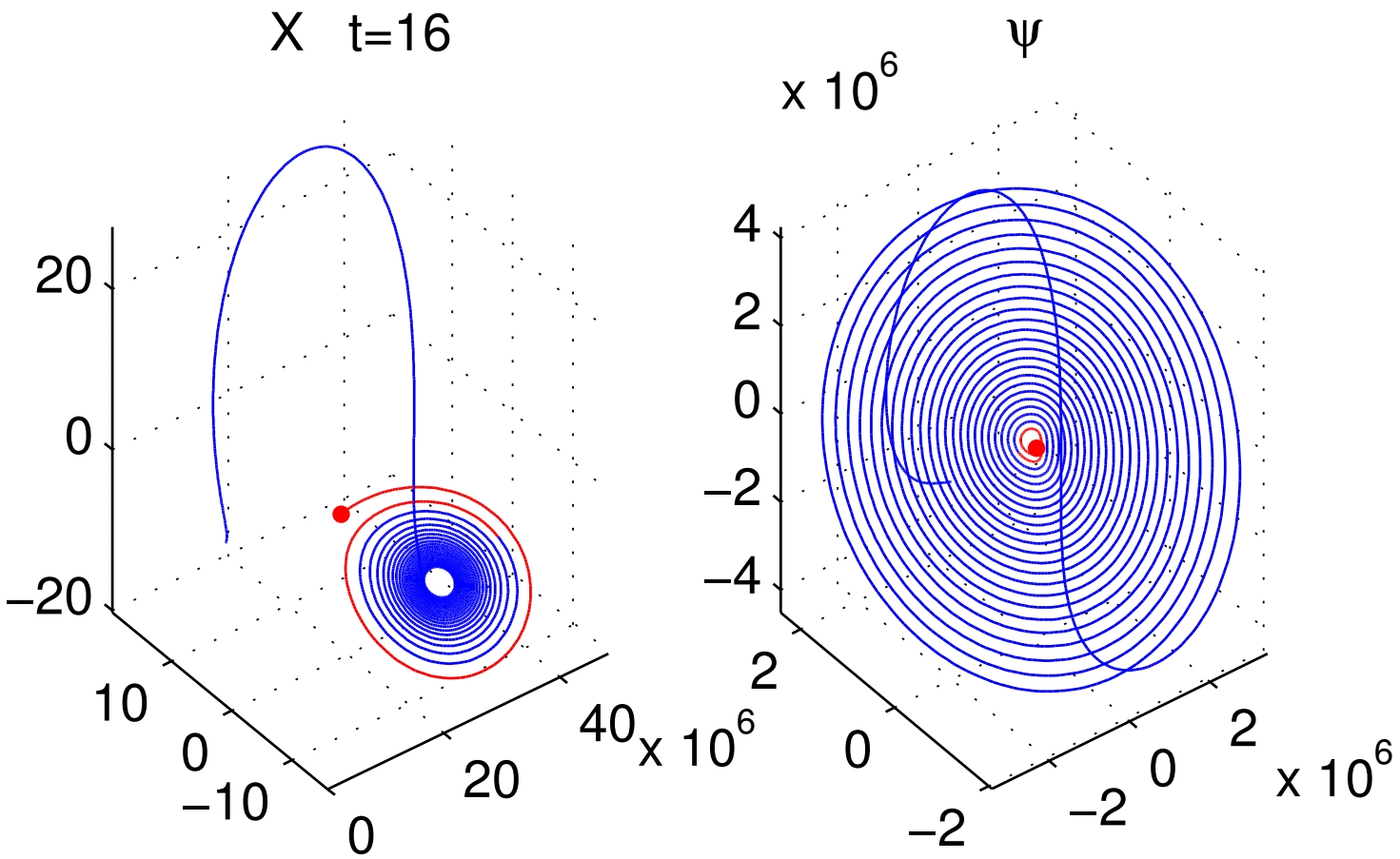
Lorenz with Dual



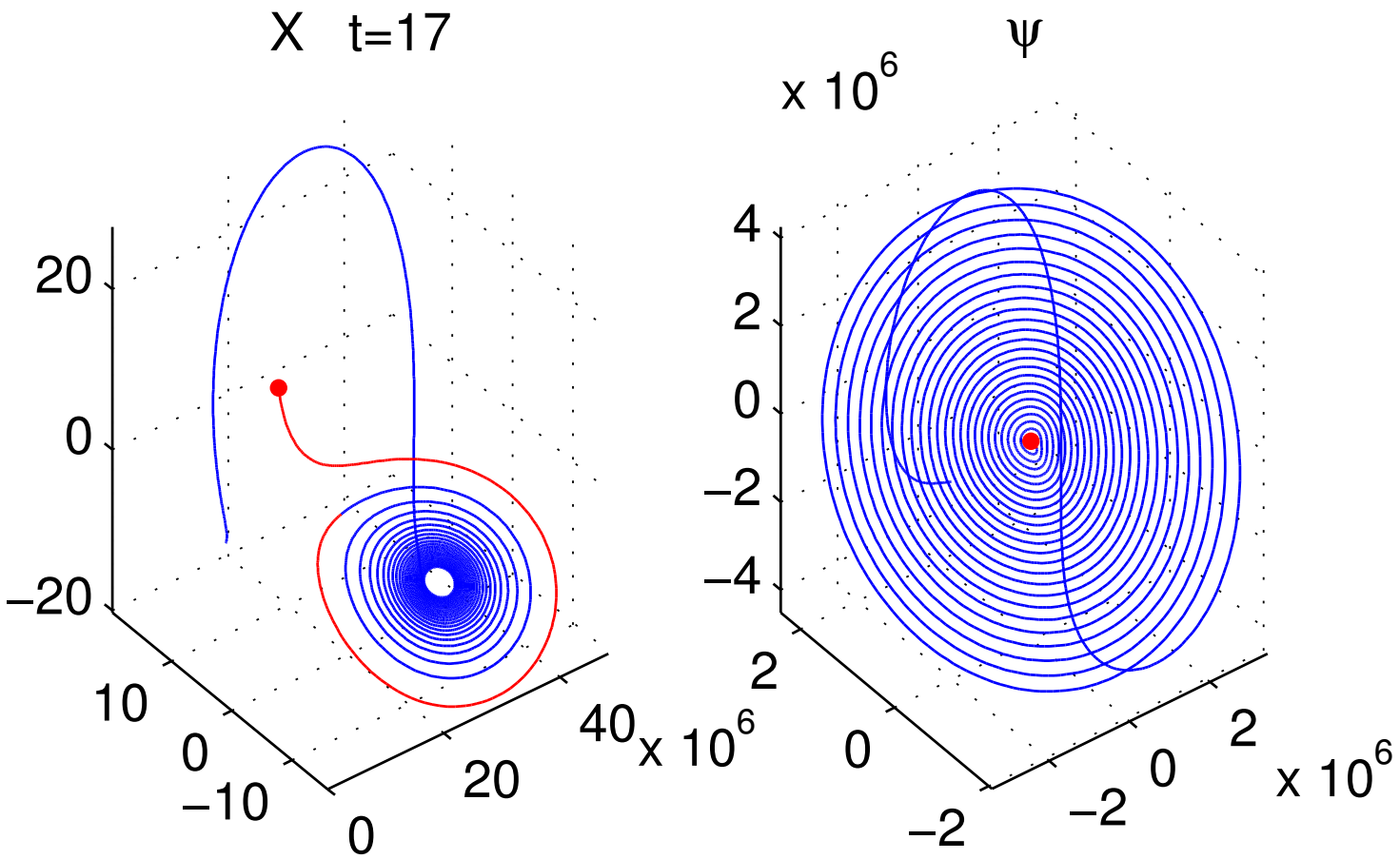
Lorenz with Dual



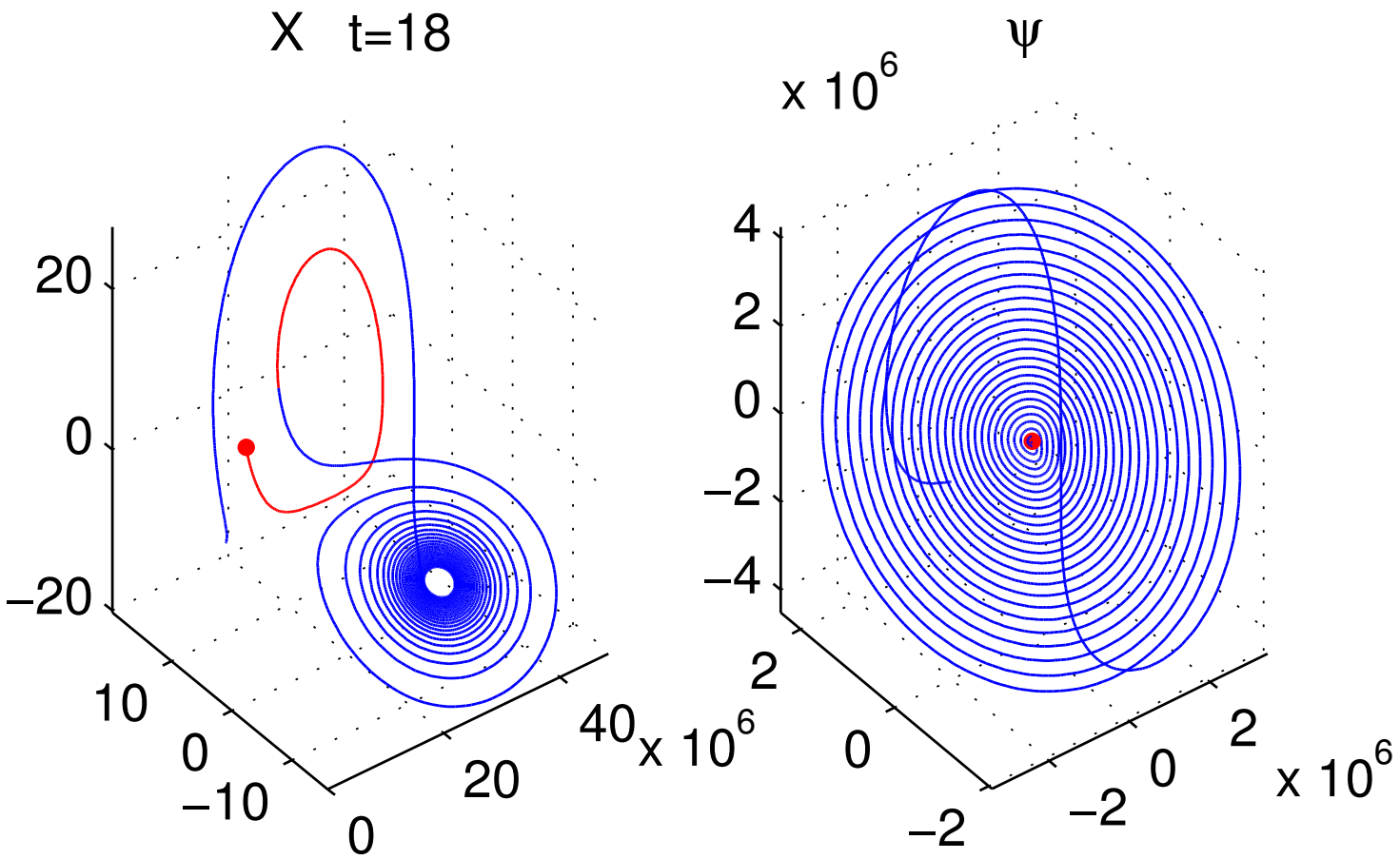
Lorenz with Dual



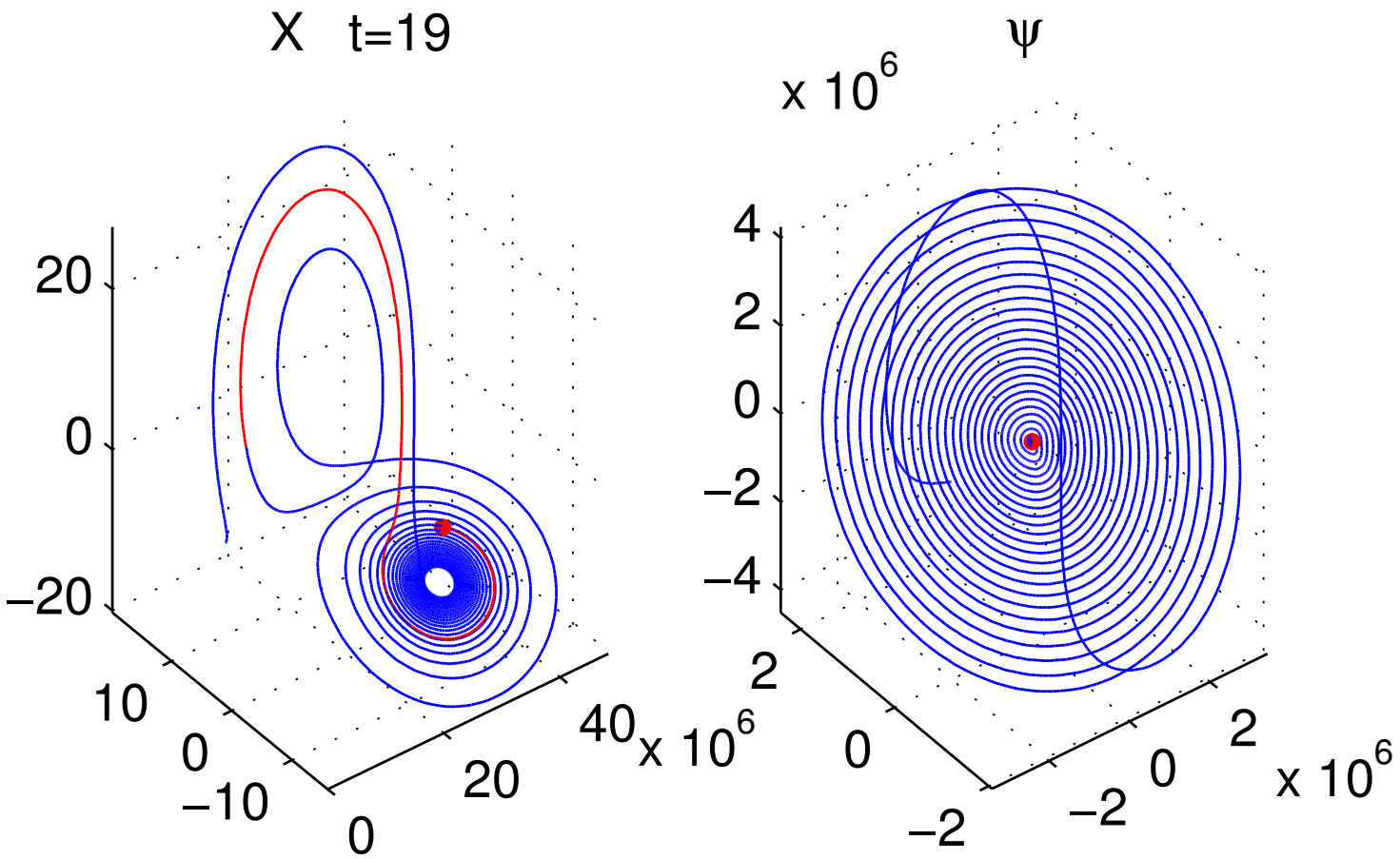
Lorenz with Dual



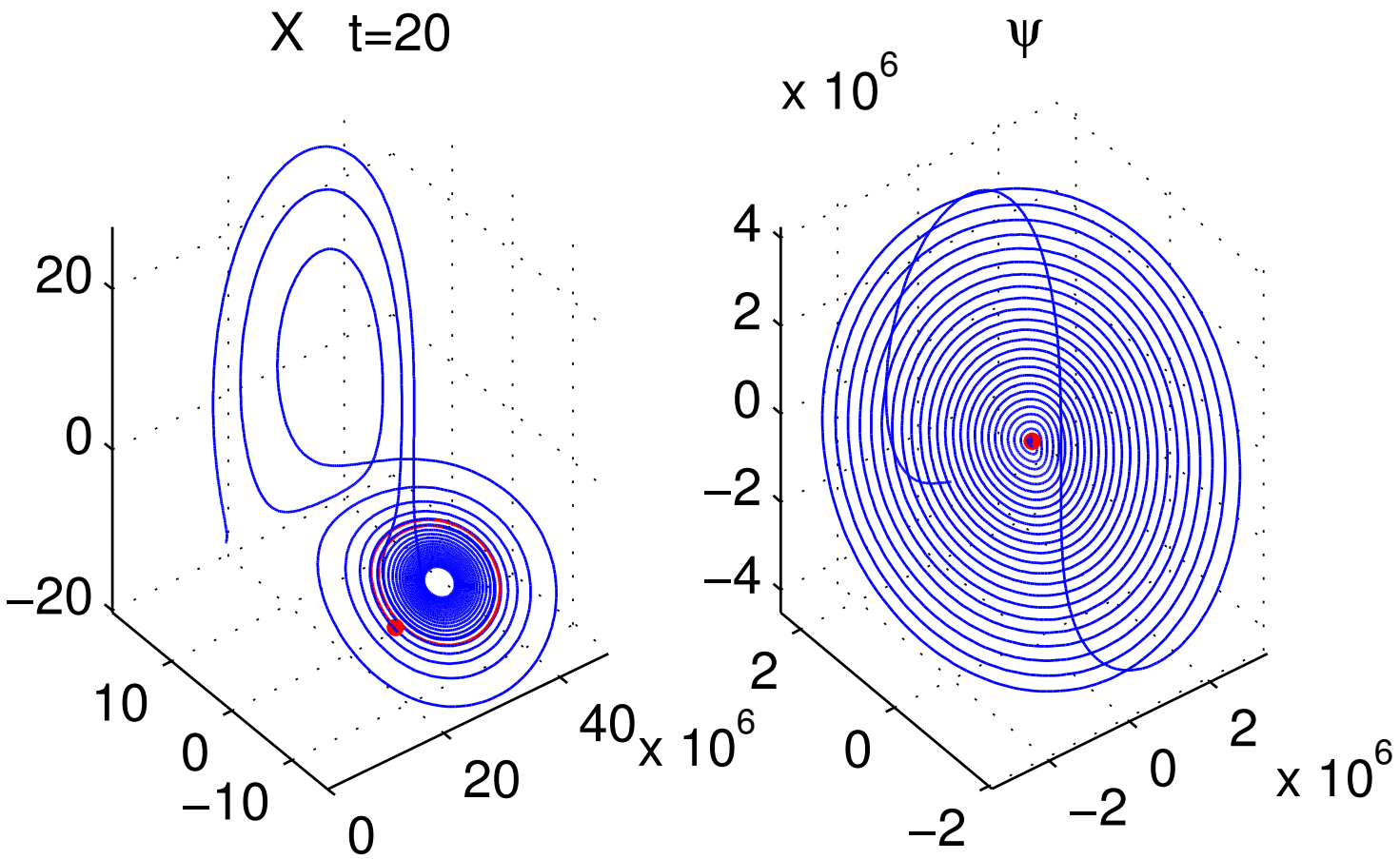
Lorenz with Dual



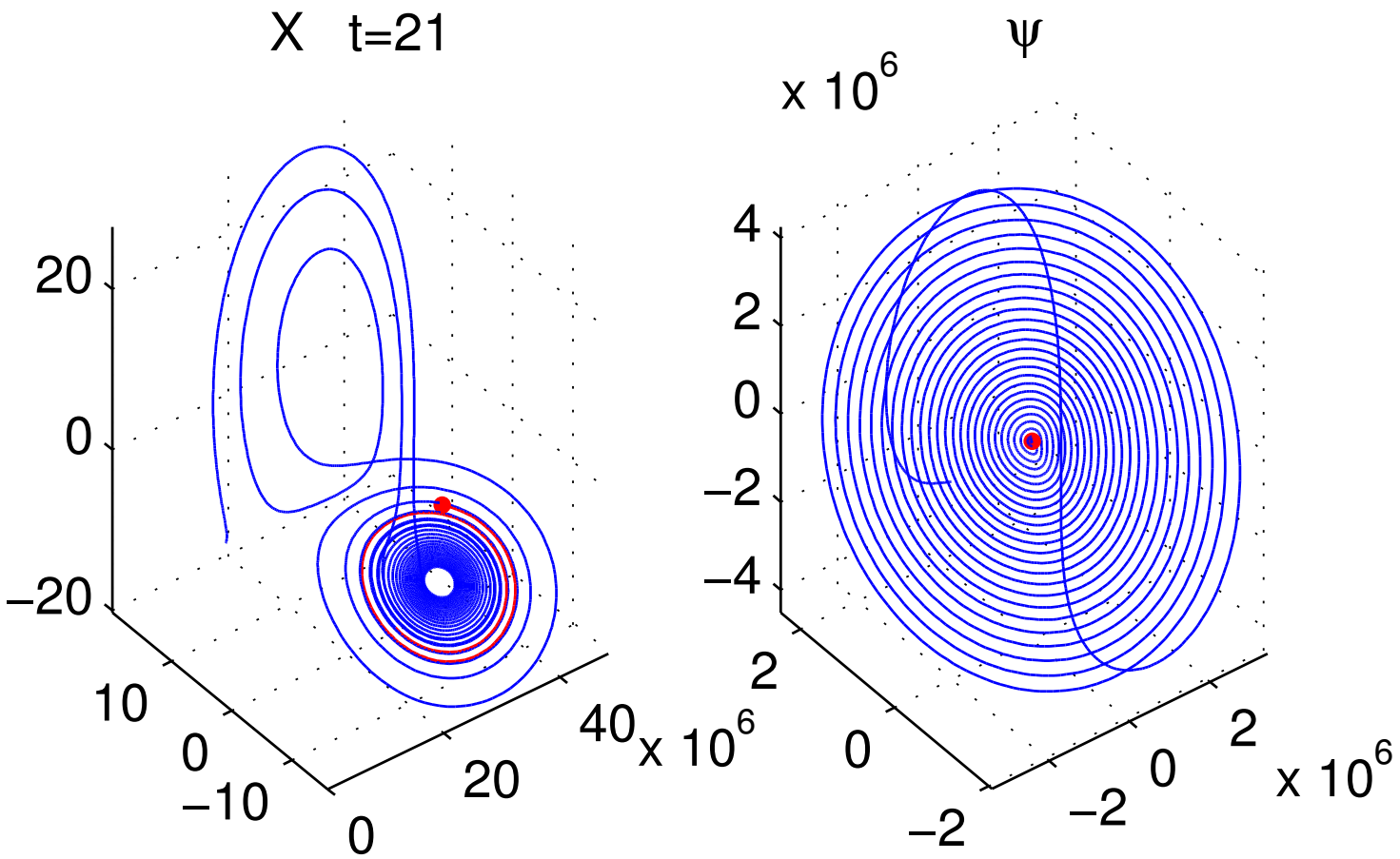
Lorenz with Dual



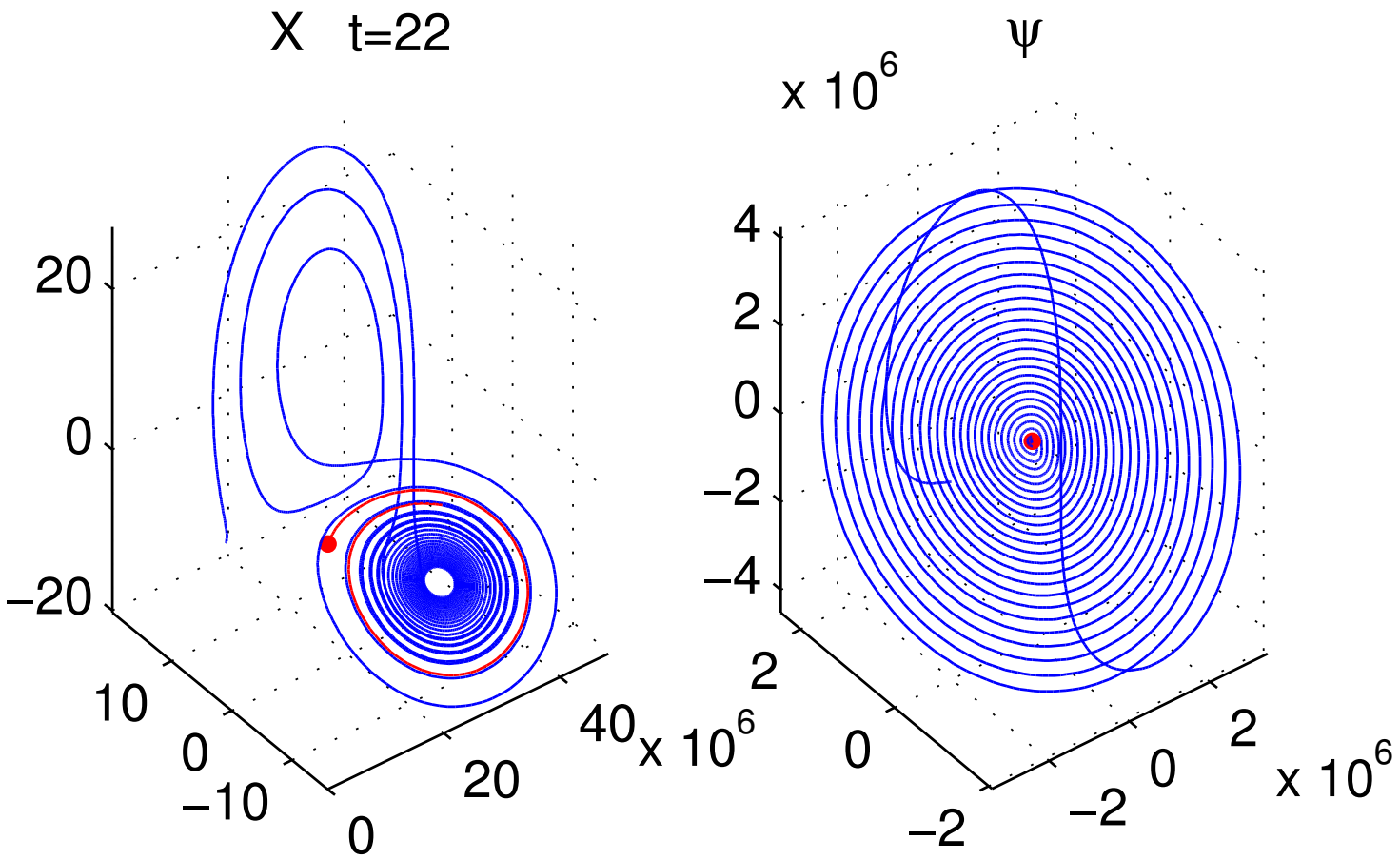
Lorenz with Dual



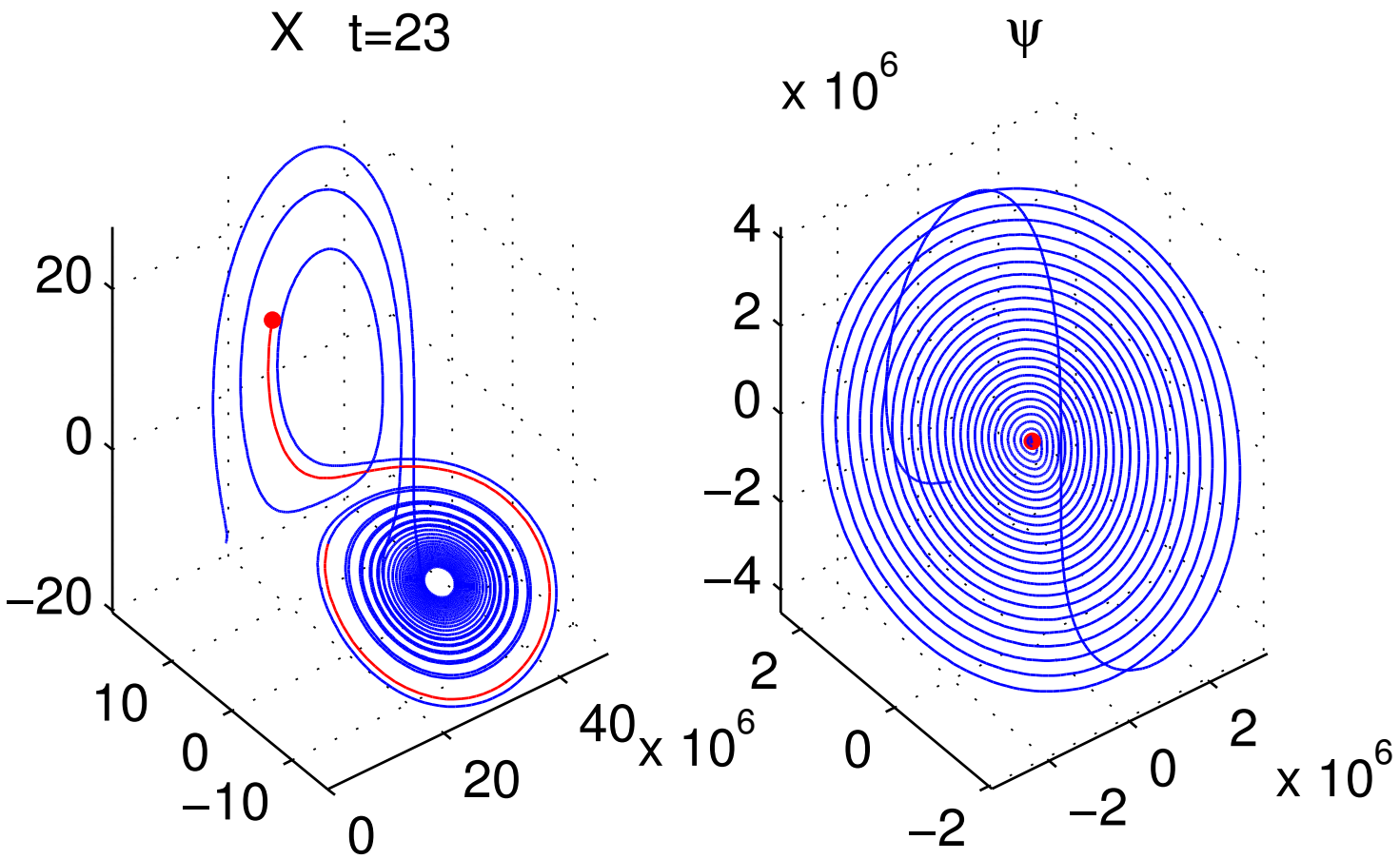
Lorenz with Dual



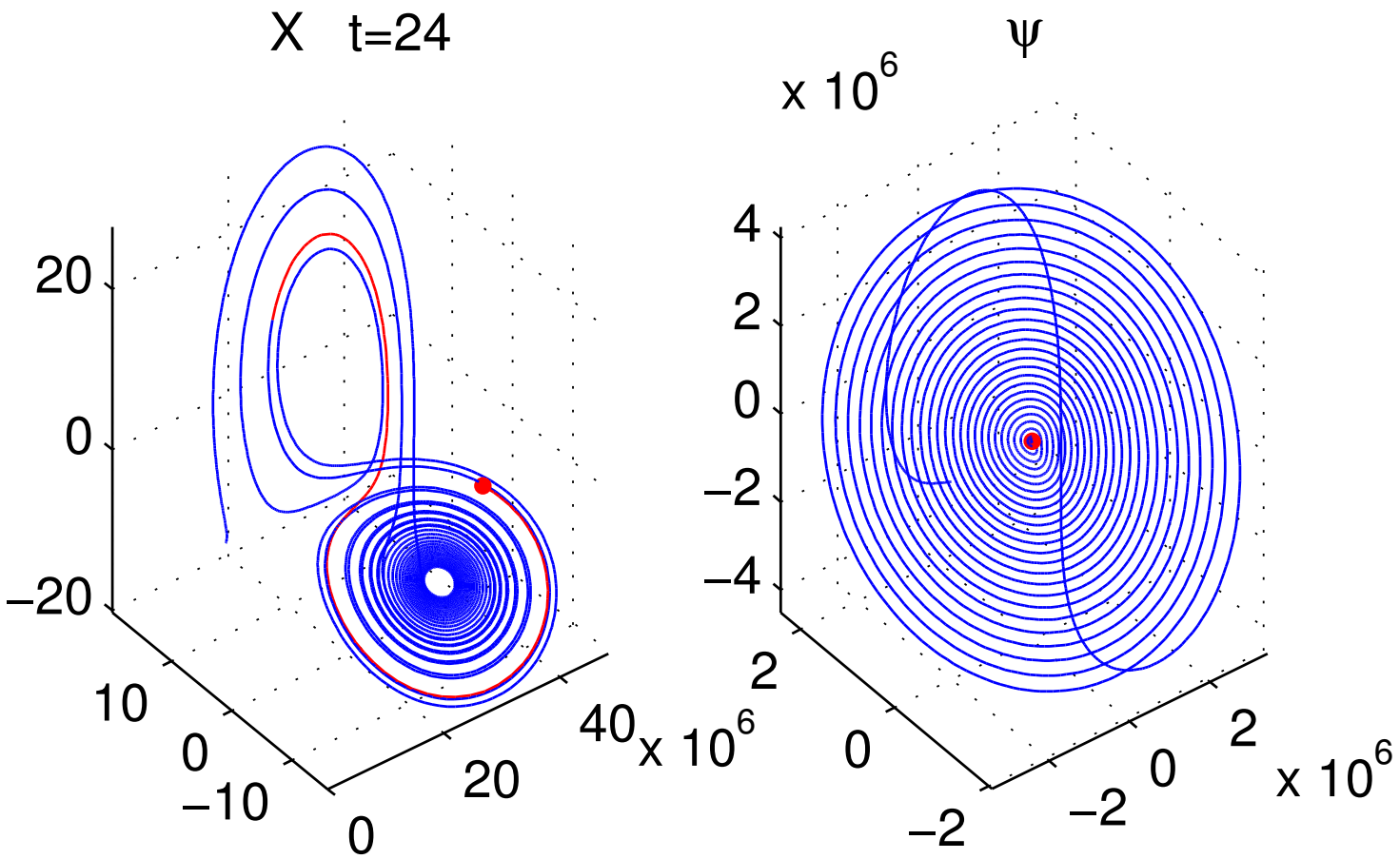
Lorenz with Dual



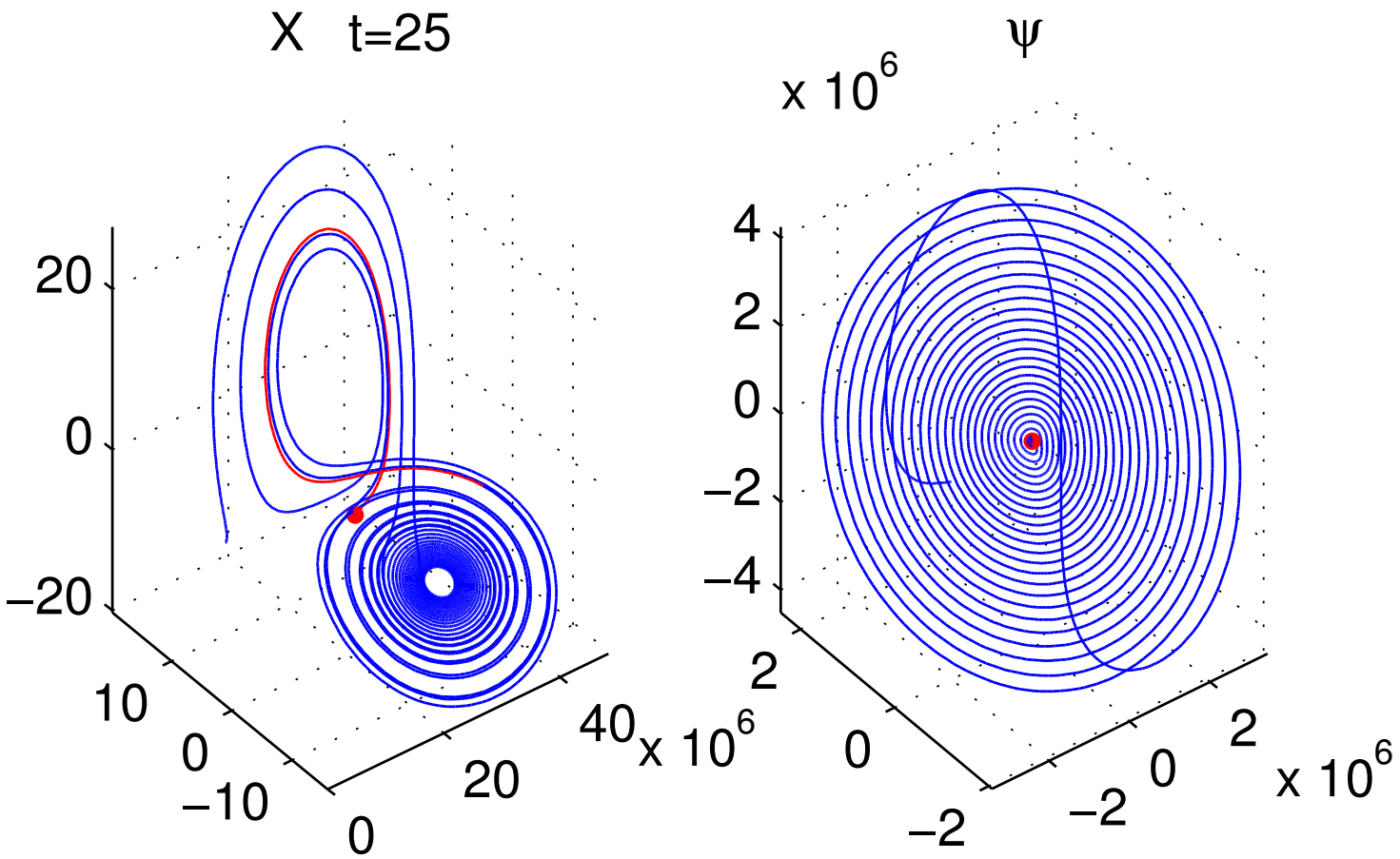
Lorenz with Dual



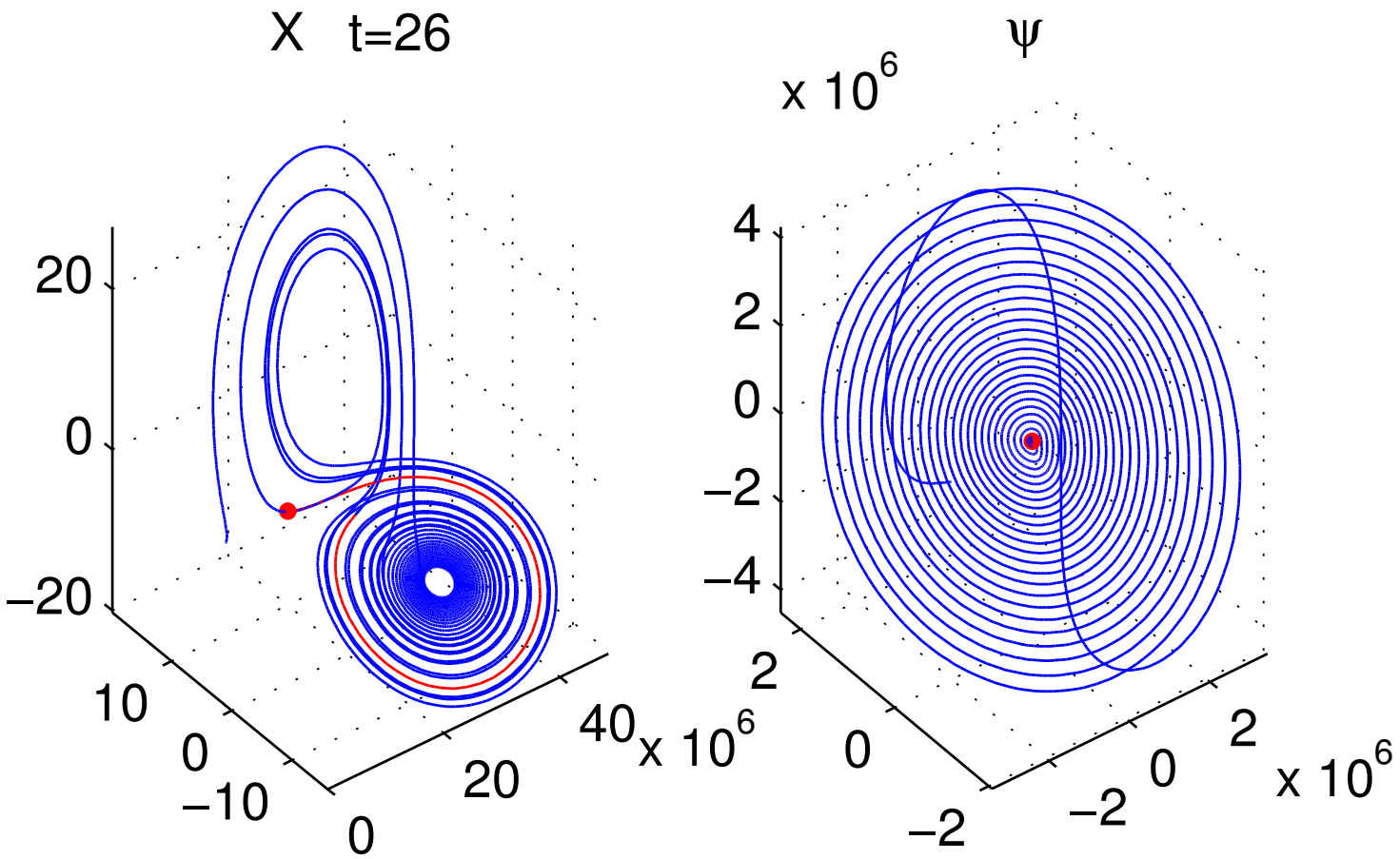
Lorenz with Dual



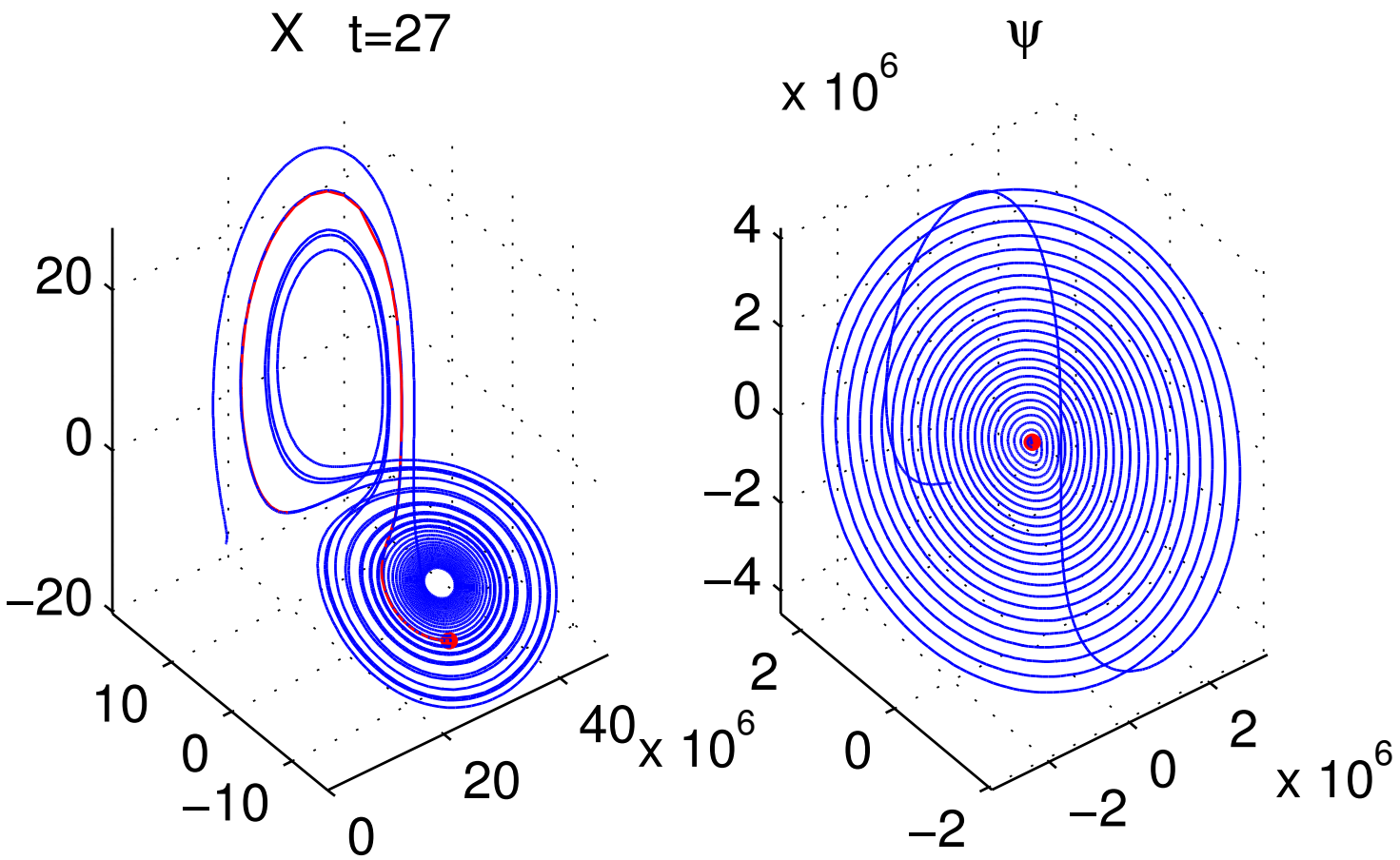
Lorenz with Dual



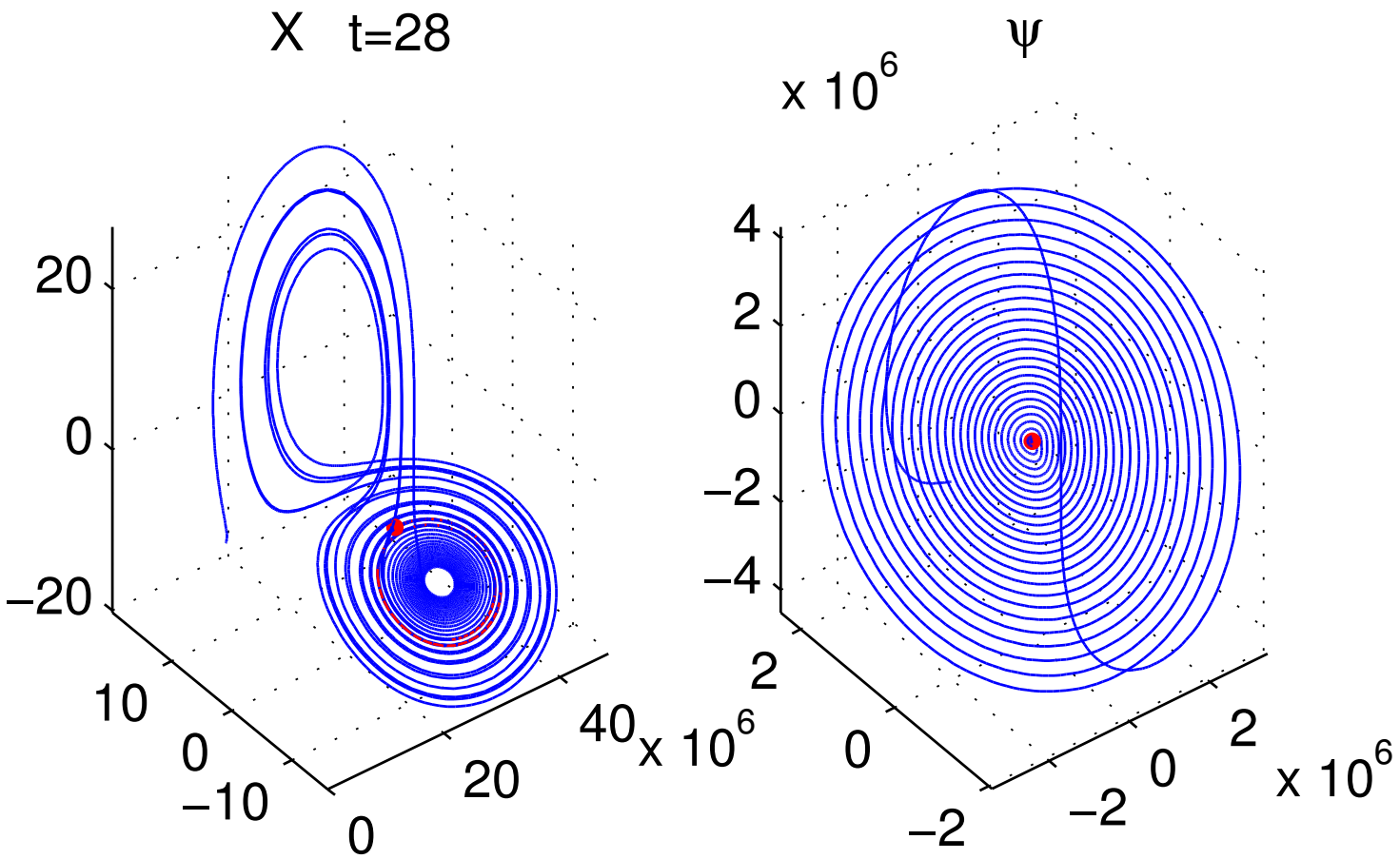
Lorenz with Dual



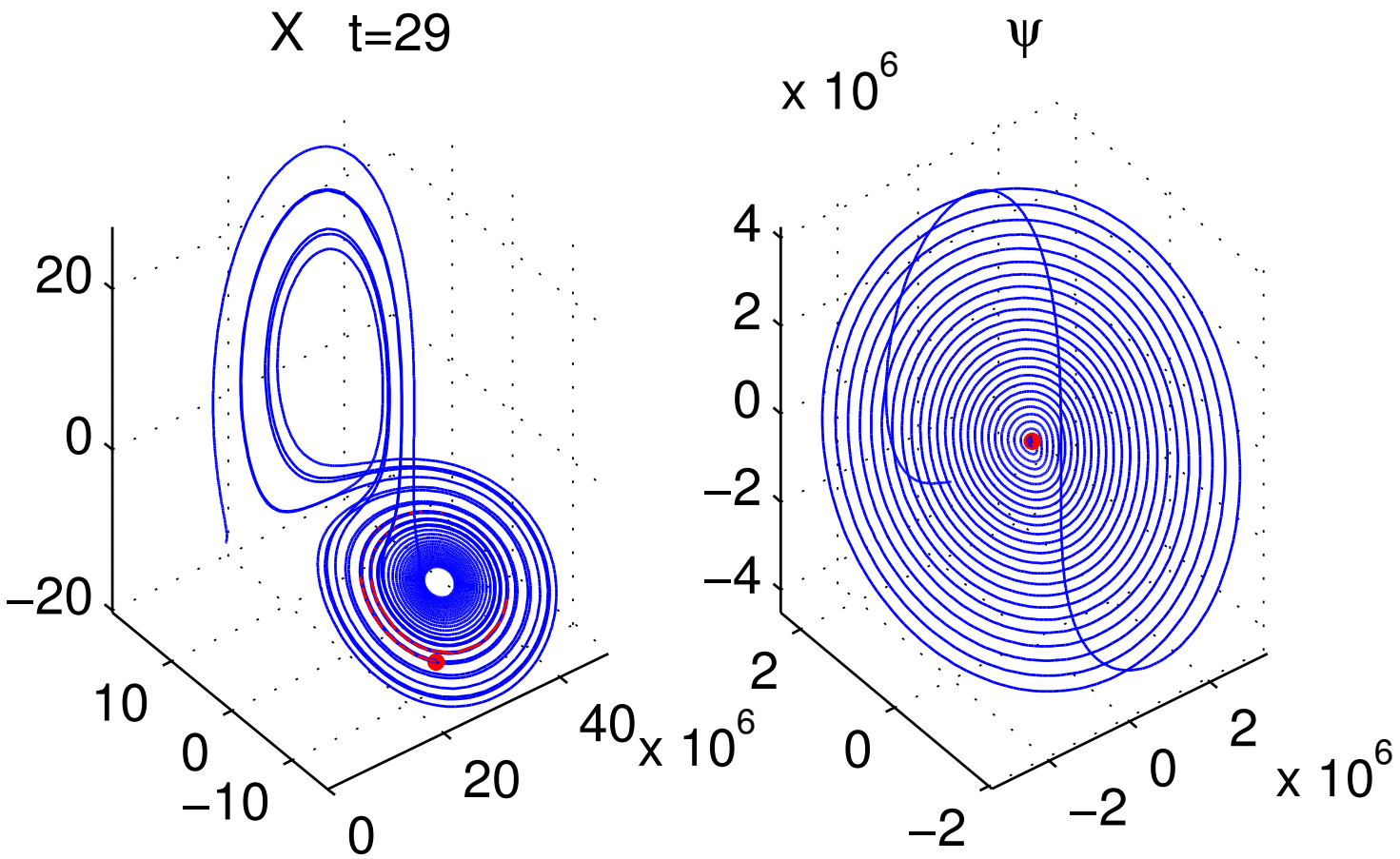
Lorenz with Dual



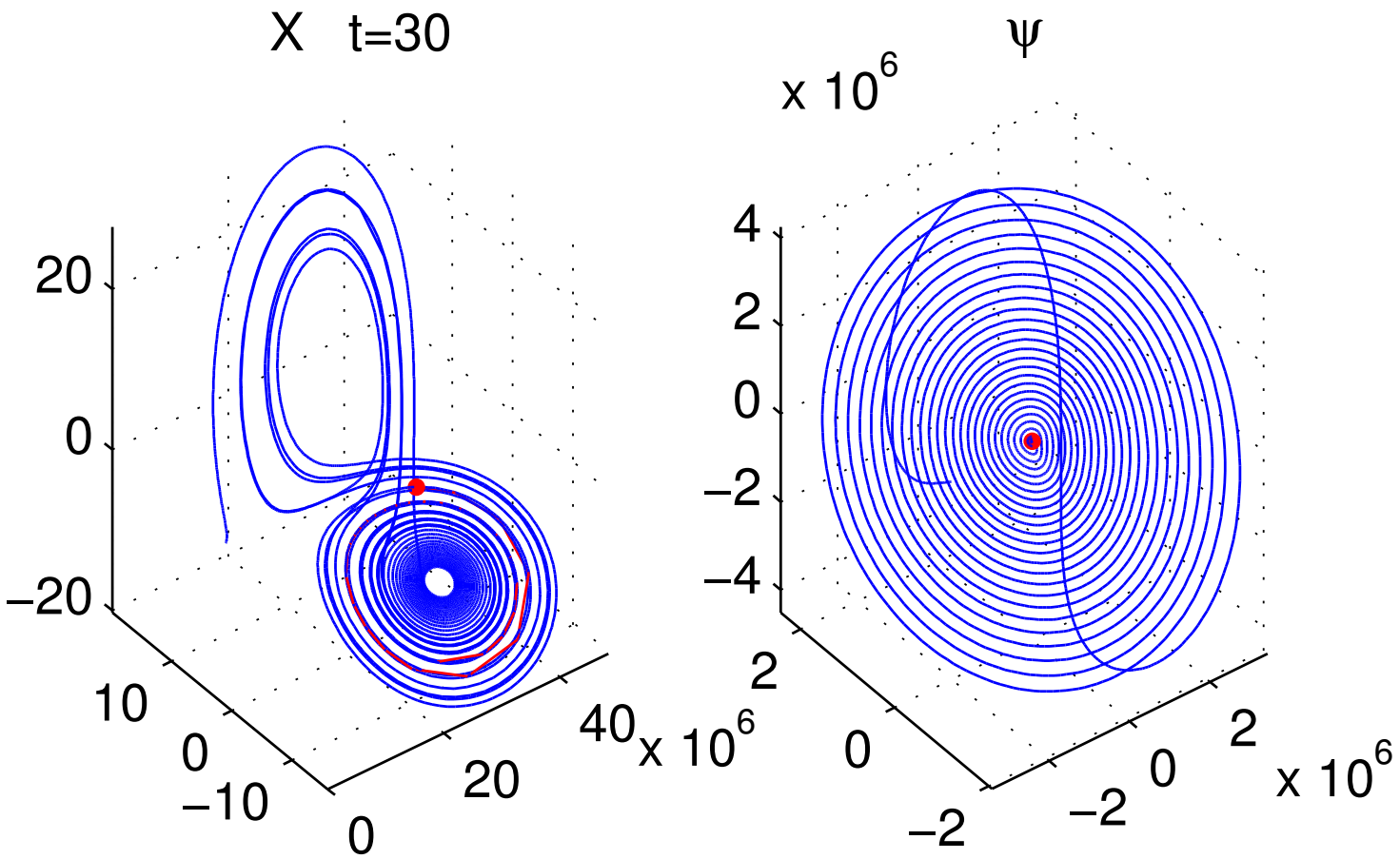
Lorenz with Dual



Lorenz with Dual



Lorenz with Dual



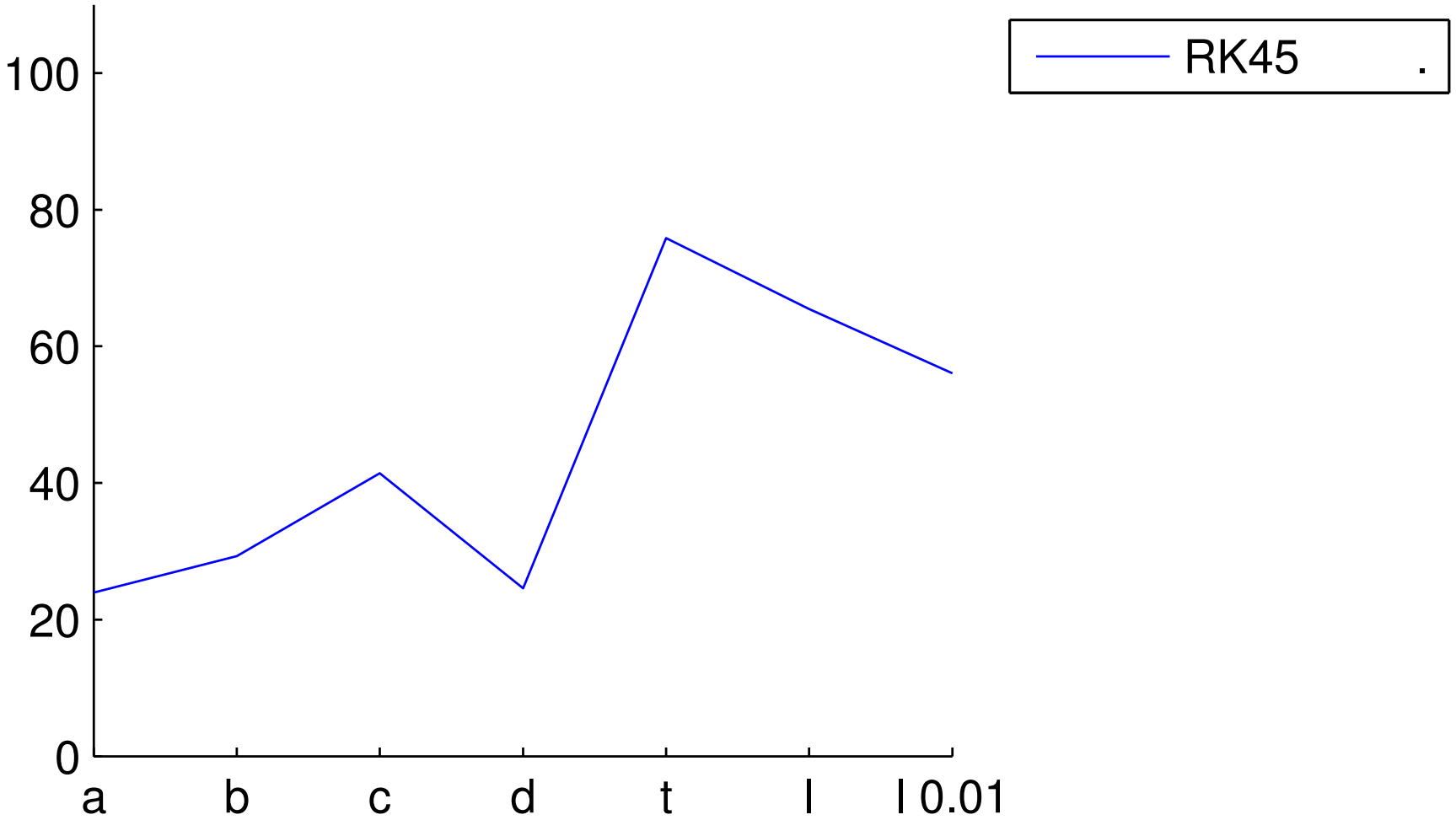
Methods implemented

- The brut method. Divide the timesteps by 2 when $r_i > \frac{\text{TOL}}{N}$.
- The basique method. Divide the timesteps by

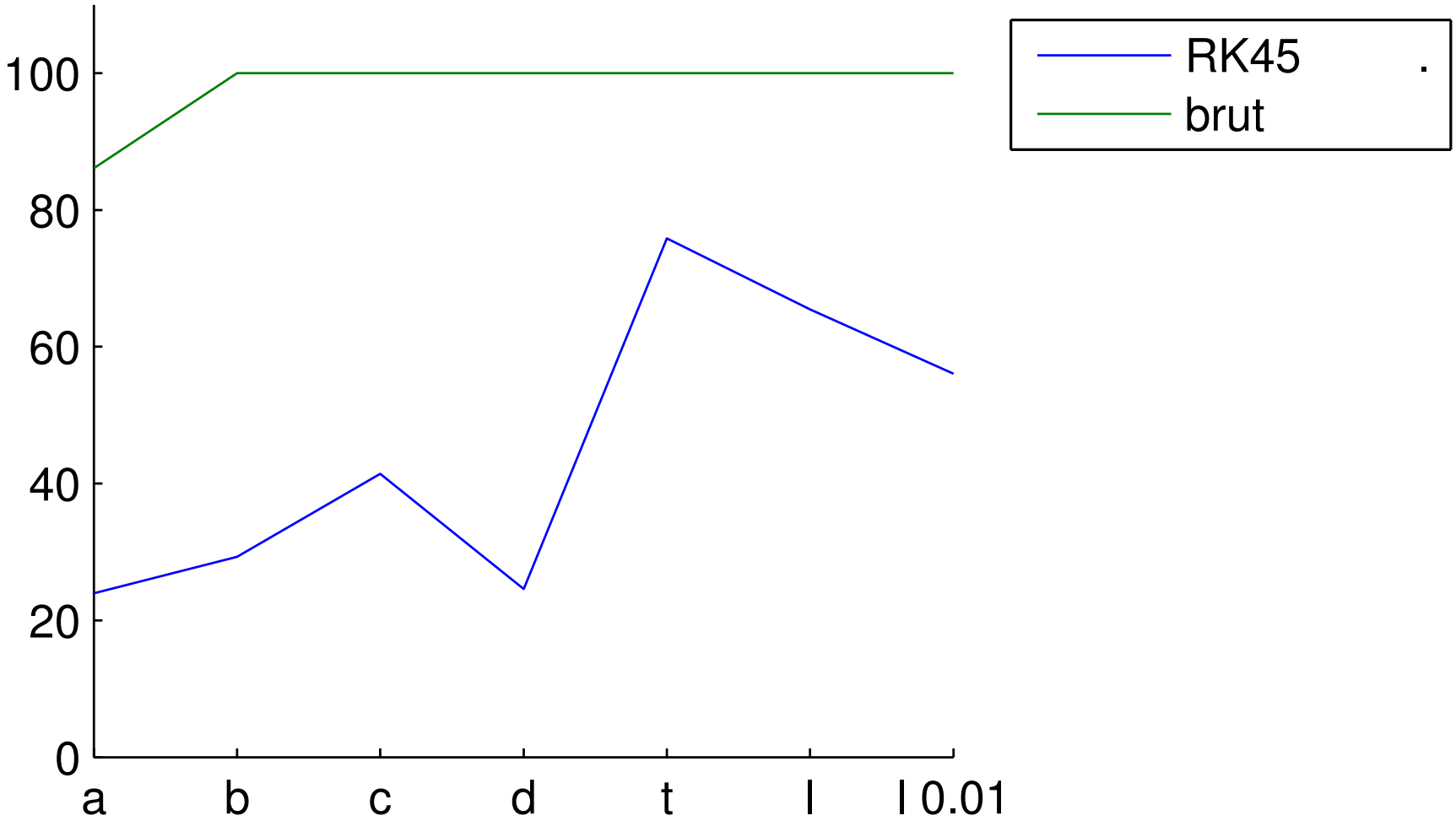
$$M = \left\lfloor \left(\frac{|r_i|}{\frac{TOL}{N}} \right)^{\frac{1}{p+1}} \right\rfloor$$

  when $r_i > \frac{\text{TOL}}{N}$.
- The demi-pas method. Same as basique, except that we estimate the error using a coarser approximation of X rather than a finer as before. The dual is solved on a mesh 2 times coarser than the primal.

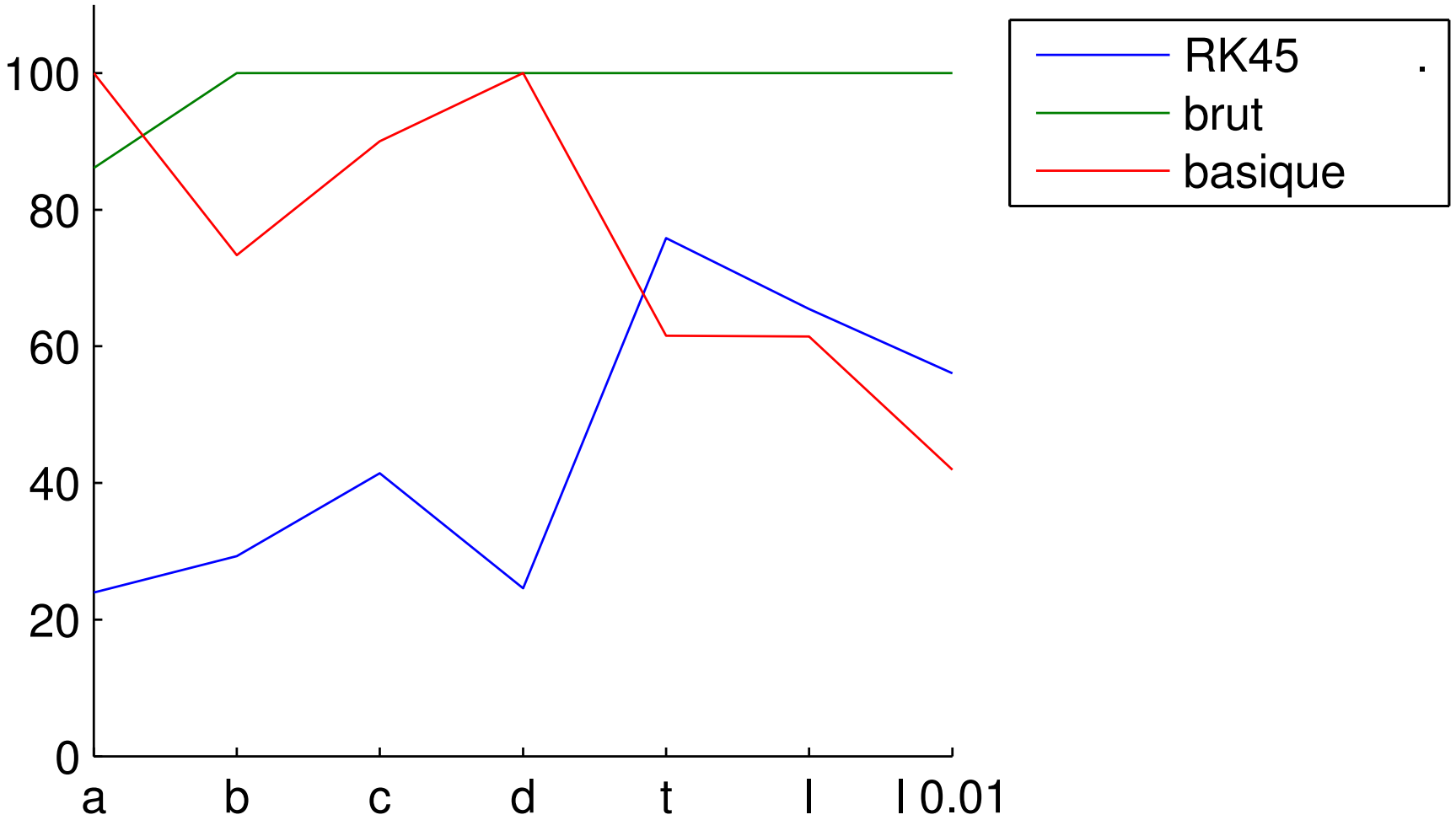
Results



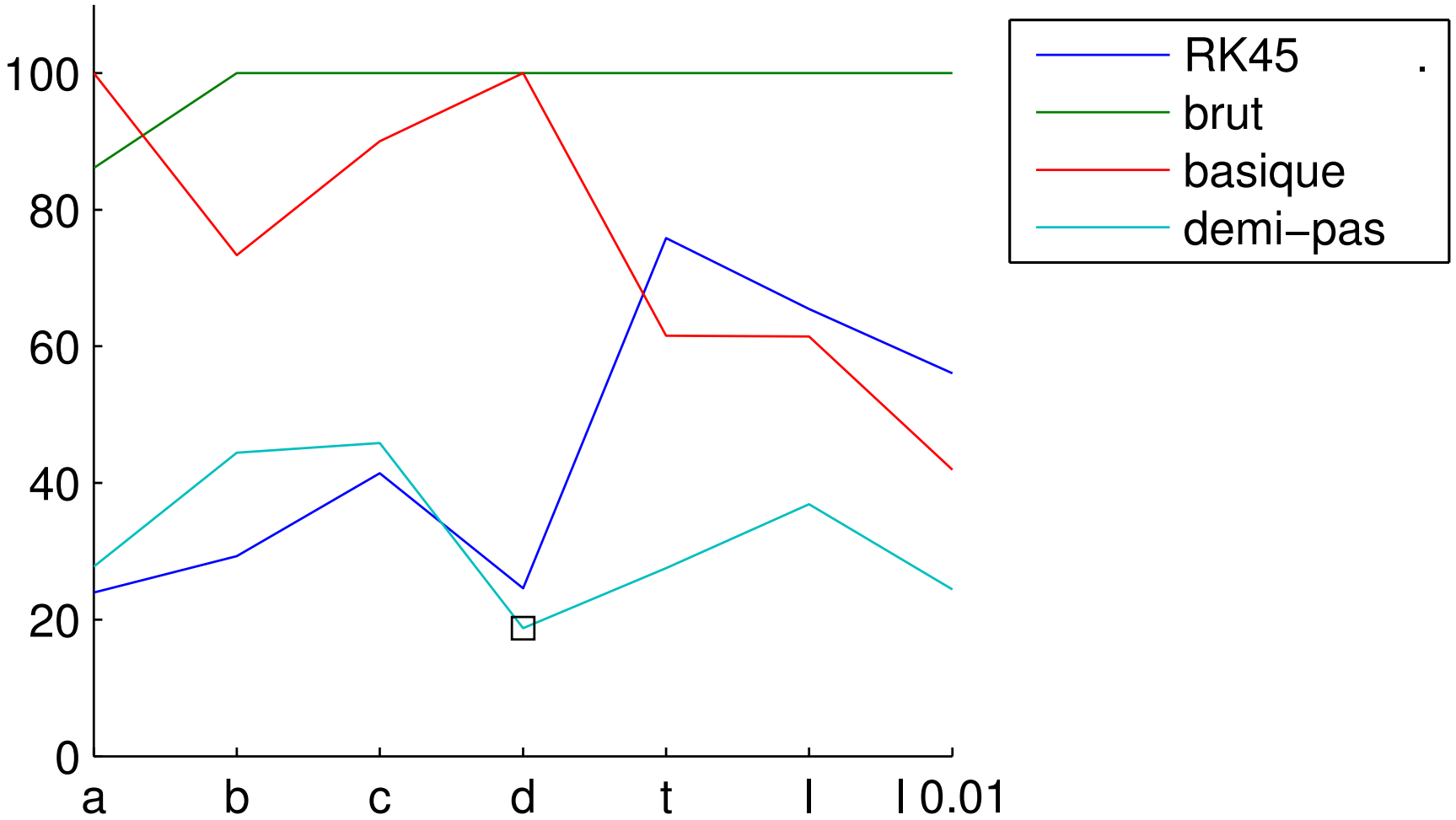
Results



Results



Results



Exponential, Non-linear with blow-up and Stiff problem

	TOL	True error	Est. errorr	# eval. a	# eval. $(a')^*$	CPU	N	it
brut	1e-08	-1.16e-09	-1.16e-09	2790	930	0.89	80	5
basique	1e-08	-3.86e-10	-3.86e-10	3240	1080	0.59	100	4
demi-pas	1e-08	-4.8e-09	-4.38e-09	900	300	0.17	30	3
RK45	1e-08	-1.33e-09	-	776	0	0.93	79	2

The Exponential

	TOL	True error	Est. errorr	# eval. a	# eval. $(a')^*$	CPU	N	it
brut	0.1	0.0125	0.00629	810	270	0.18	16	4
basique	0.1	0.0125	0.00629	594	198	0.12	16	3
demi-pas	0.1	0.0194	-0.0122	360	120	0.07	8	3
RK45	0.1	0.0986	-	237	0	0.06	13	3

Non-linear with blow-up

	TOL	True error	Est. errorr	# eval. a	# eval. $(a')^*$	CPU	N	it
brut	1e-08	1.39e-09	1.41e-09	2160	720	0.41	45	5
basique	1e-08	4.77e-10	4.87e-10	1944	648	0.35	53	3
demi-pas	1e-08	3.68e-10	8.41e-10	990	330	0.18	50	2
RK45	1e-08	5.92e-10	-	895	0	0.13	146	1

Stiff problem

Singularity and Transition to Turbulence

	TOL	True error	Est. errorr	# eval. a	# eval. $(a')^*$	CPU	N	it
brut	0.1	0.0484	0.0976	4320	1440	0.81	24	16
basique	0.1	0.0484	0.0976	4320	1440	0.79	24	16
demi-pas	0.1	-1.1	0.0858	810	270	0.16	10	6
RK45	0.1	0.0055	-	1061	0	0.19	71	5

Singularity

	TOL	True error	Est. error	# eval. a	# eval. $(a')^*$	CPU	N	it
brut	1e-06	5.49e-08	1.55e-07	184716	61572	44.82	2531	7
basique	1e-06	6.31e-08	1.7e-07	113652	37884	27.4	2514	4
demi-pas	1e-06	9.16e-08	3.19e-07	50832	16944	12.23	1335	3
RK45	1e-06	-9.69e-07	-	140045	0	24.04	9569	5

Transition to turbulence

Lorenz, 2 tolerances

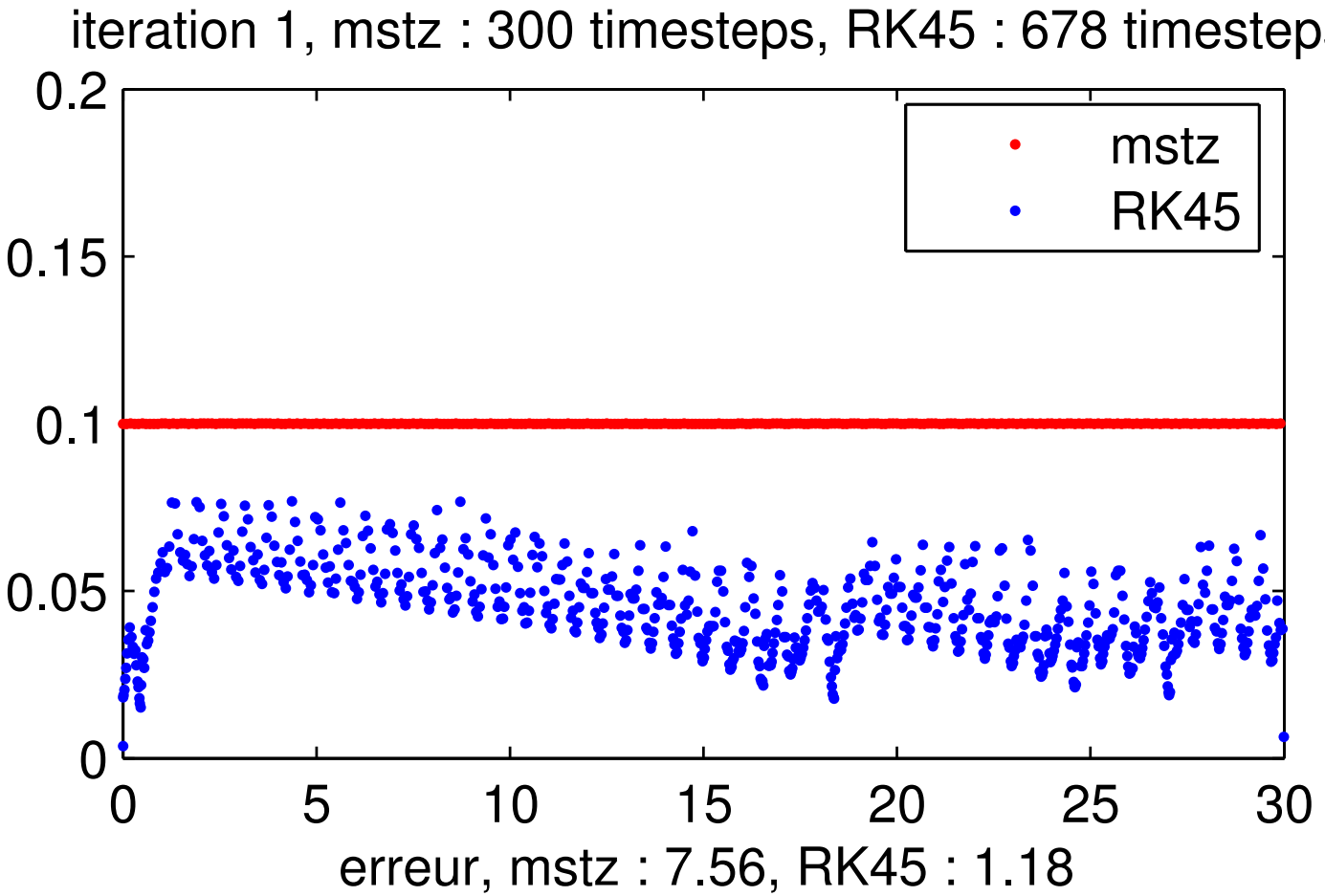
	TOL	True error	Est. error	# eval. a	# eval. $(a')^*$	CPU	N	it
brut	0.1	0.0623	0.0674	256734	85578	57.91	6461	6
basique	0.1	-0.0251	-0.0261	157680	52560	34.43	5789	3
demi-pas	0.1	-0.0146	-0.0182	94716	31572	20.7	3176	3
RK45	0.1	-0.0147	-	168110	0	25.29	10513	8

Lorenz problem

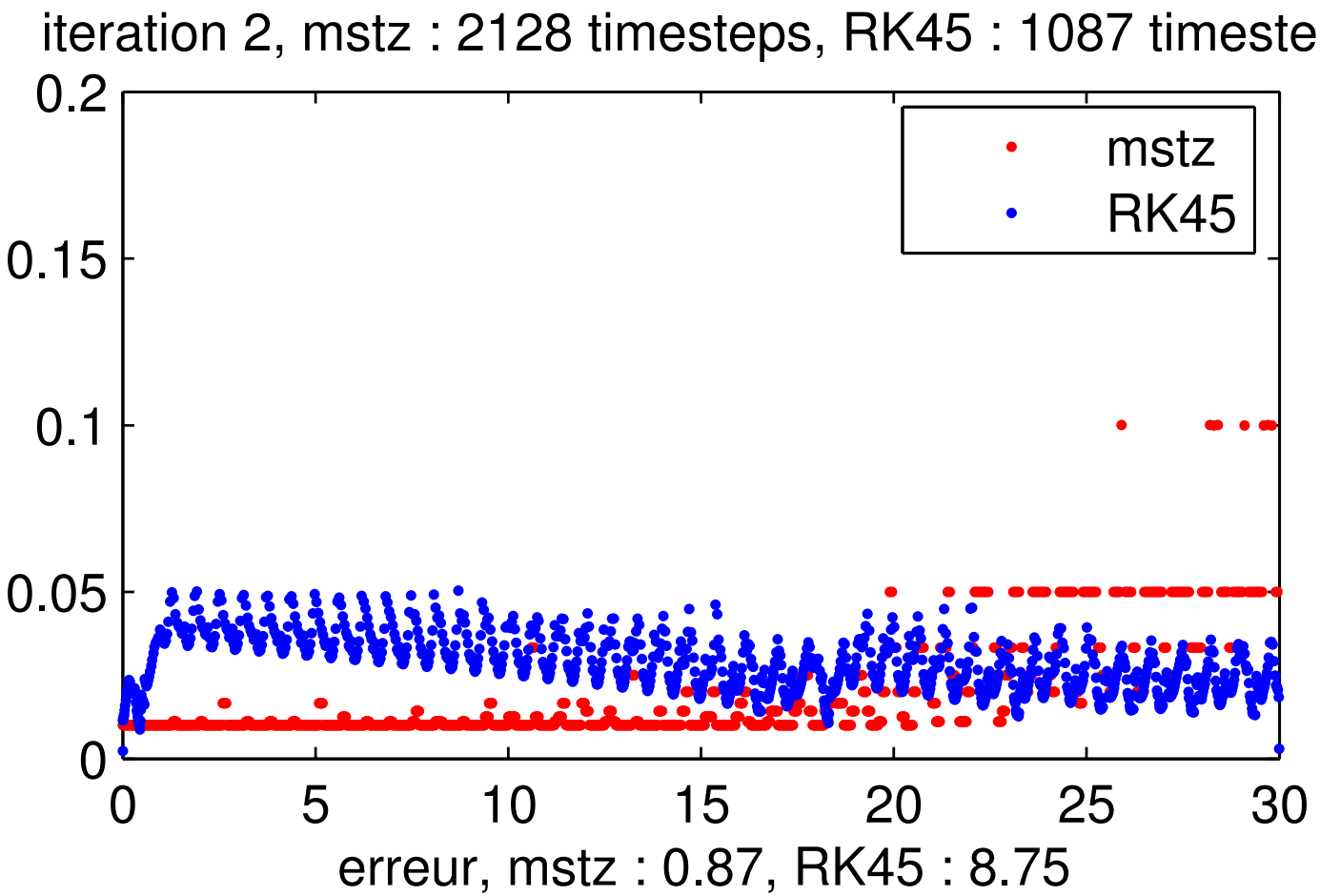
	TOL	Vraie erreur	Est. erreur	# eval. a	# eval. $(a')^*$	CPU	N	it
brut	0.01	0.000979	0.000998	472788	157596	108.75	10463	7
basique	0.01	-0.0037	-0.00365	198108	66036	44.29	7949	3
demi-pas	0.01	-0.00428	-0.00521	115434	38478	25.39	3985	3
RK45	0.01	-0.0015	-	264932	0	40.37	16651	8

Lorenz problem

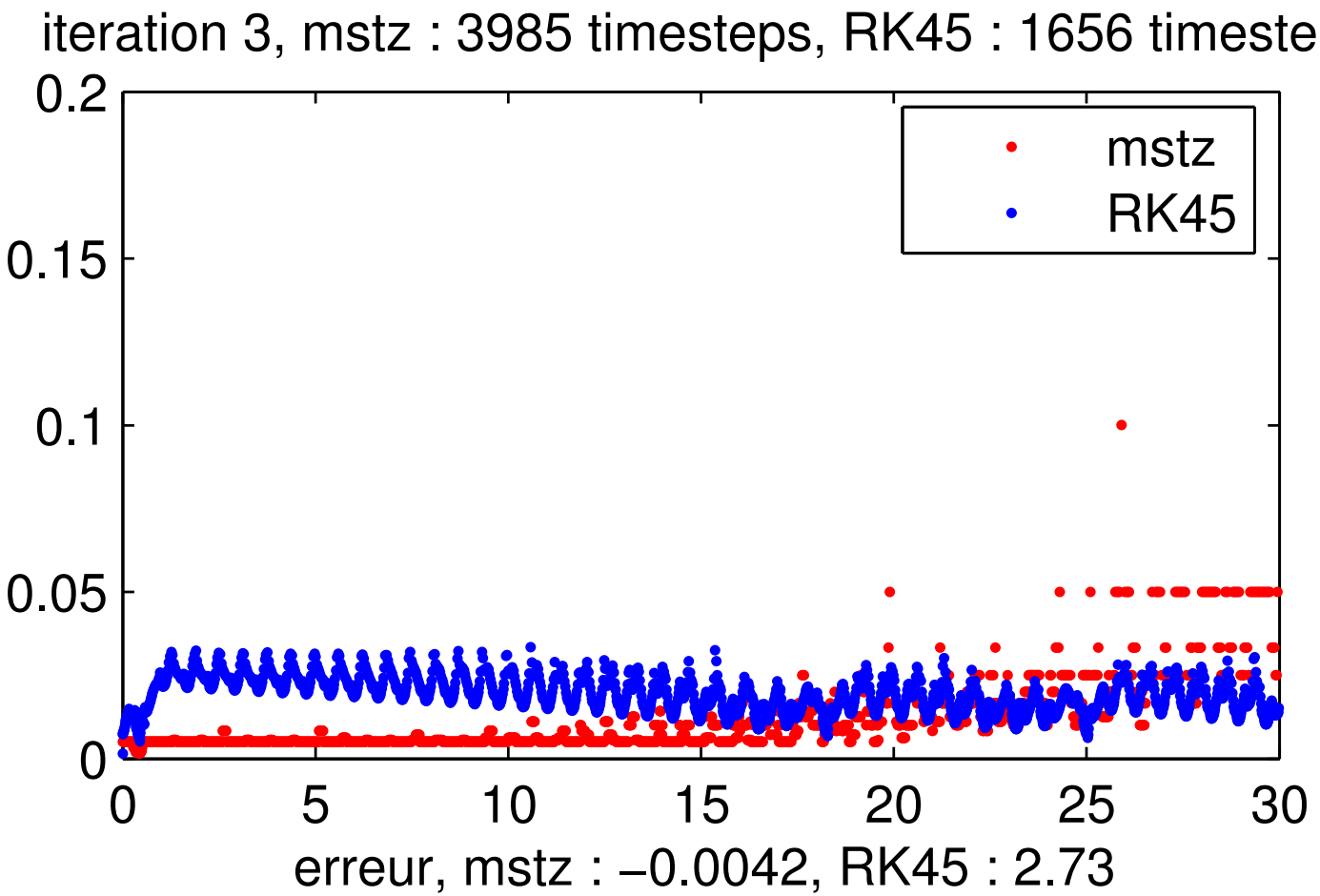
Size of time steps (Lorenz problem, TOL= 0.01)



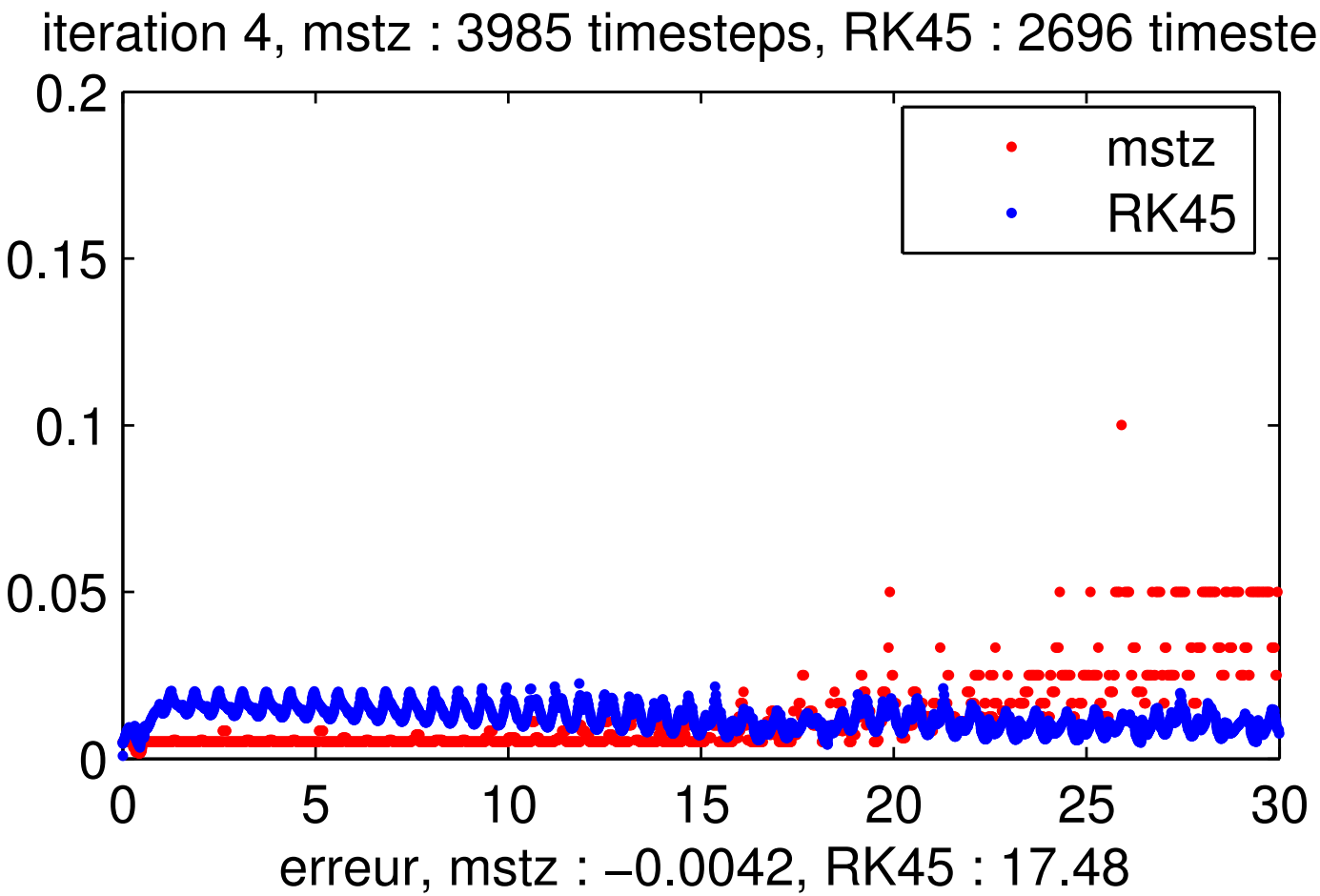
Size of time steps (Lorenz problem, TOL= 0.01)



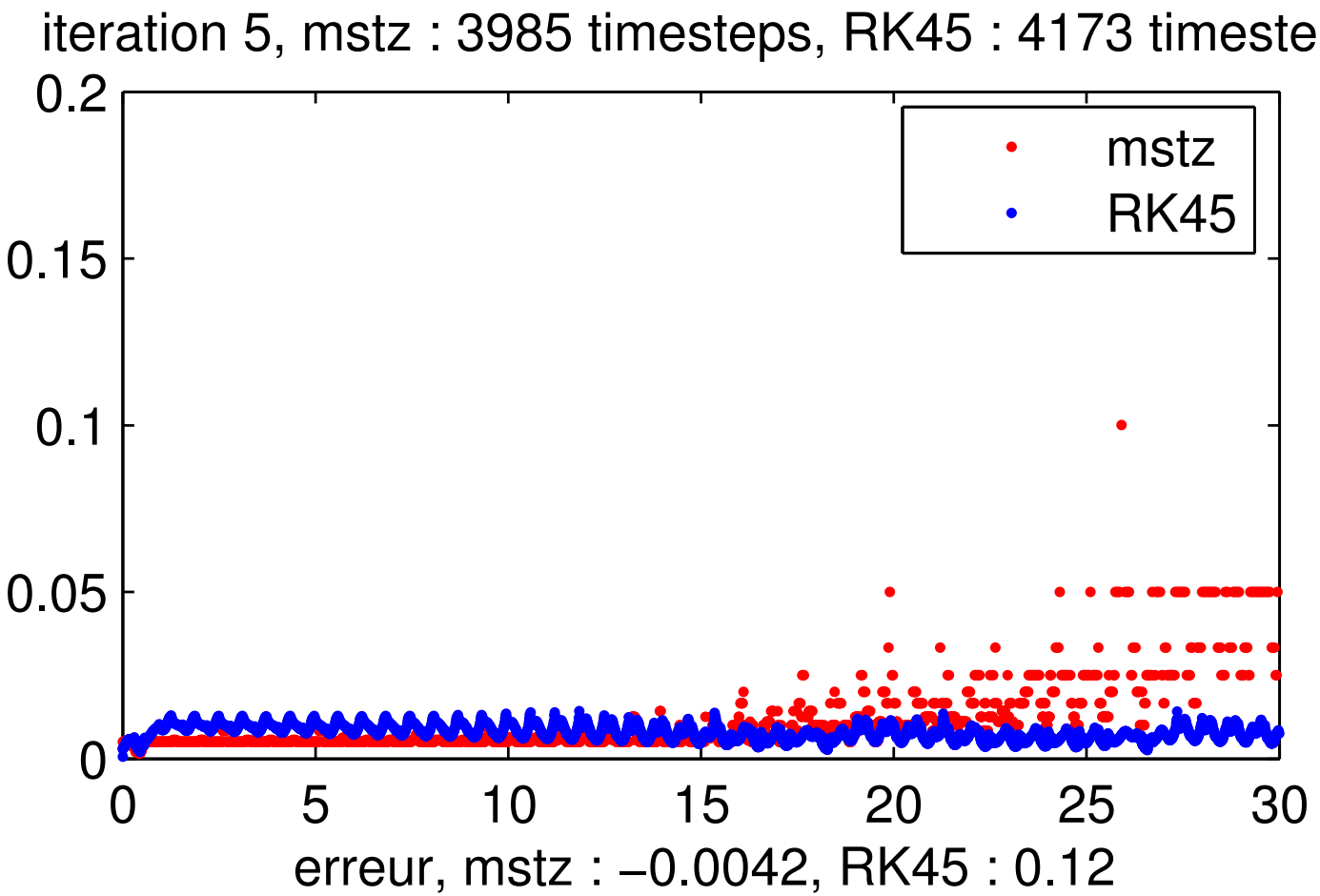
Size of time steps (Lorenz problem, TOL= 0.01)



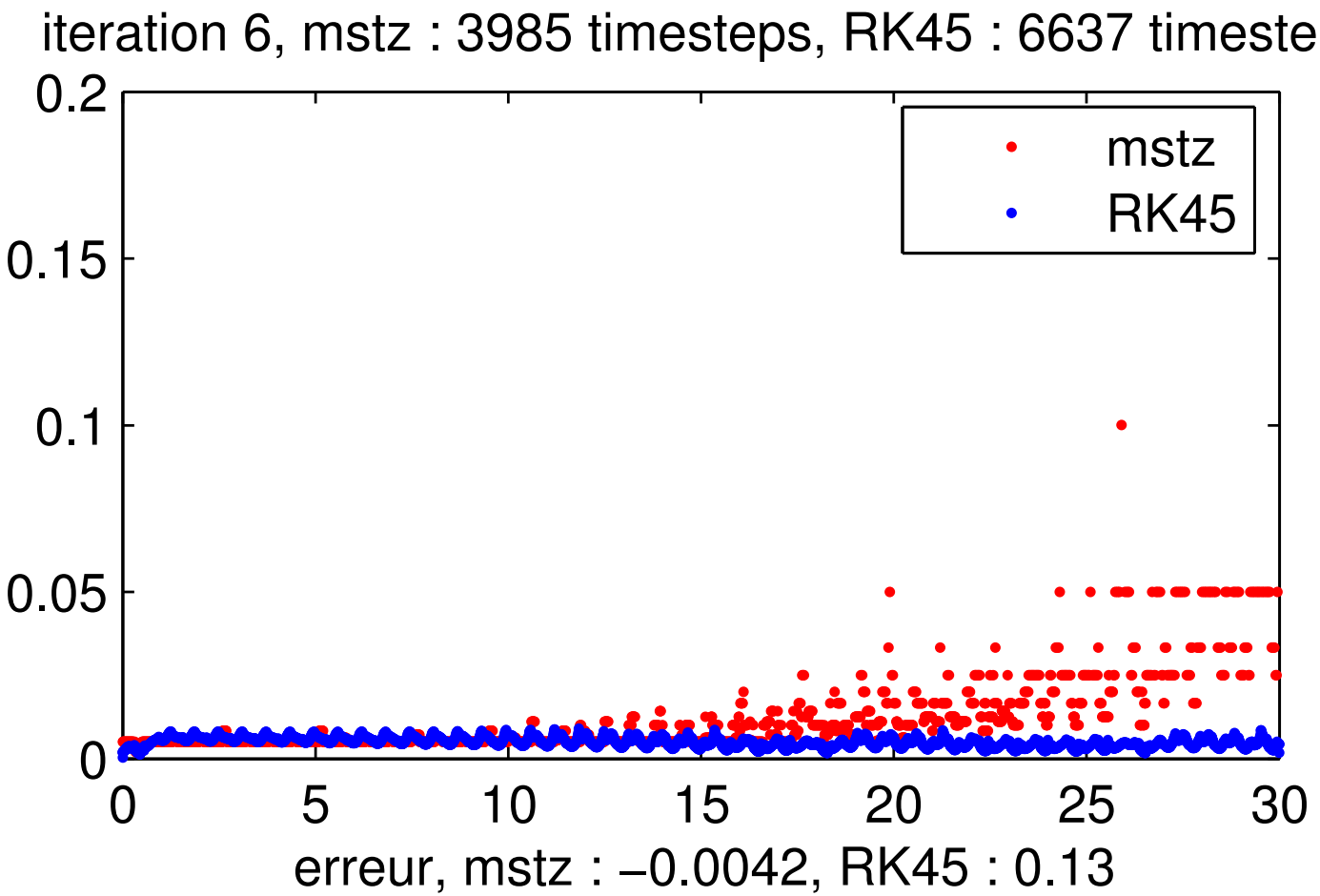
Size of time steps (Lorenz problem, TOL= 0.01)



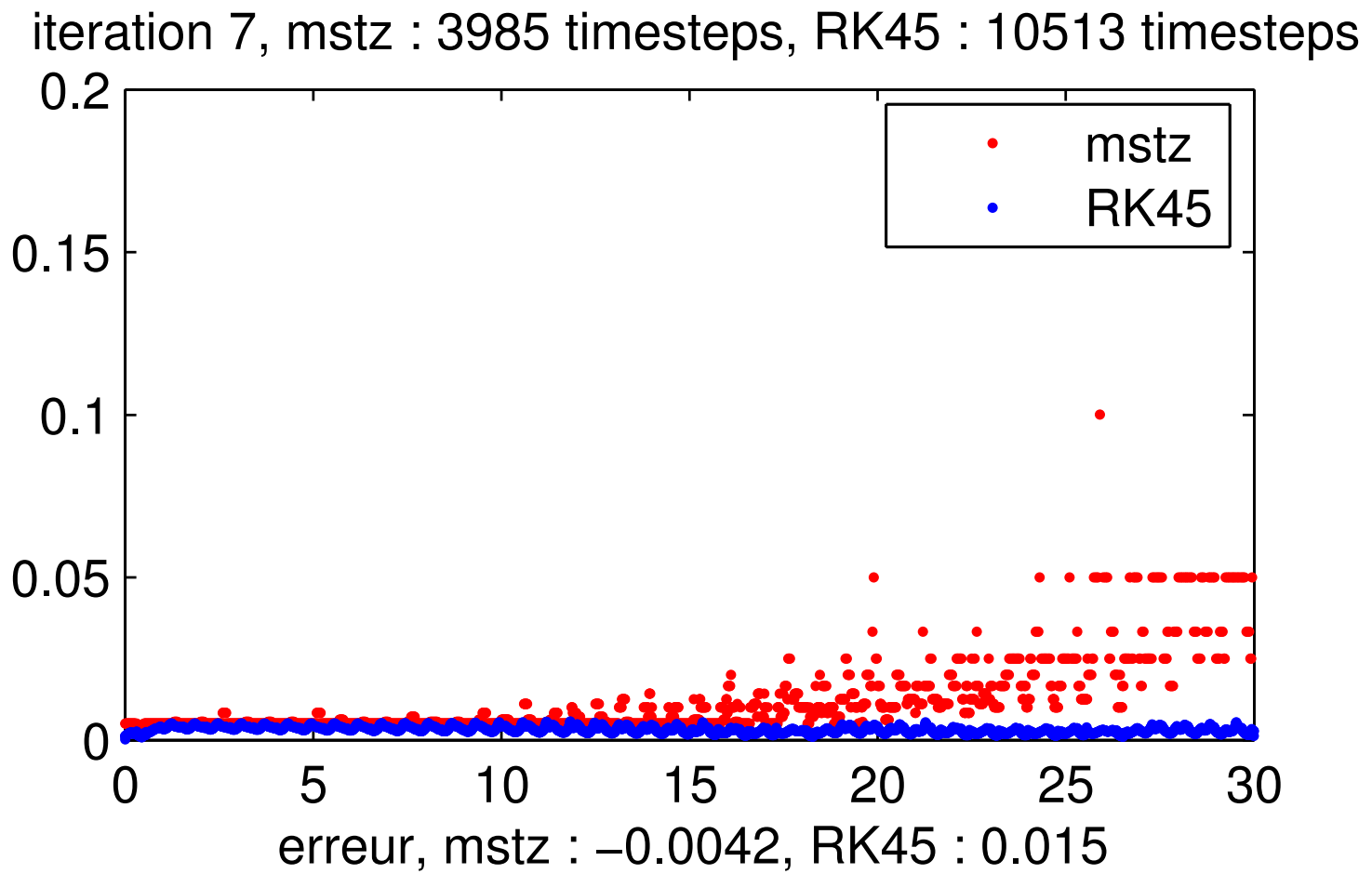
Size of time steps (Lorenz problem, TOL= 0.01)



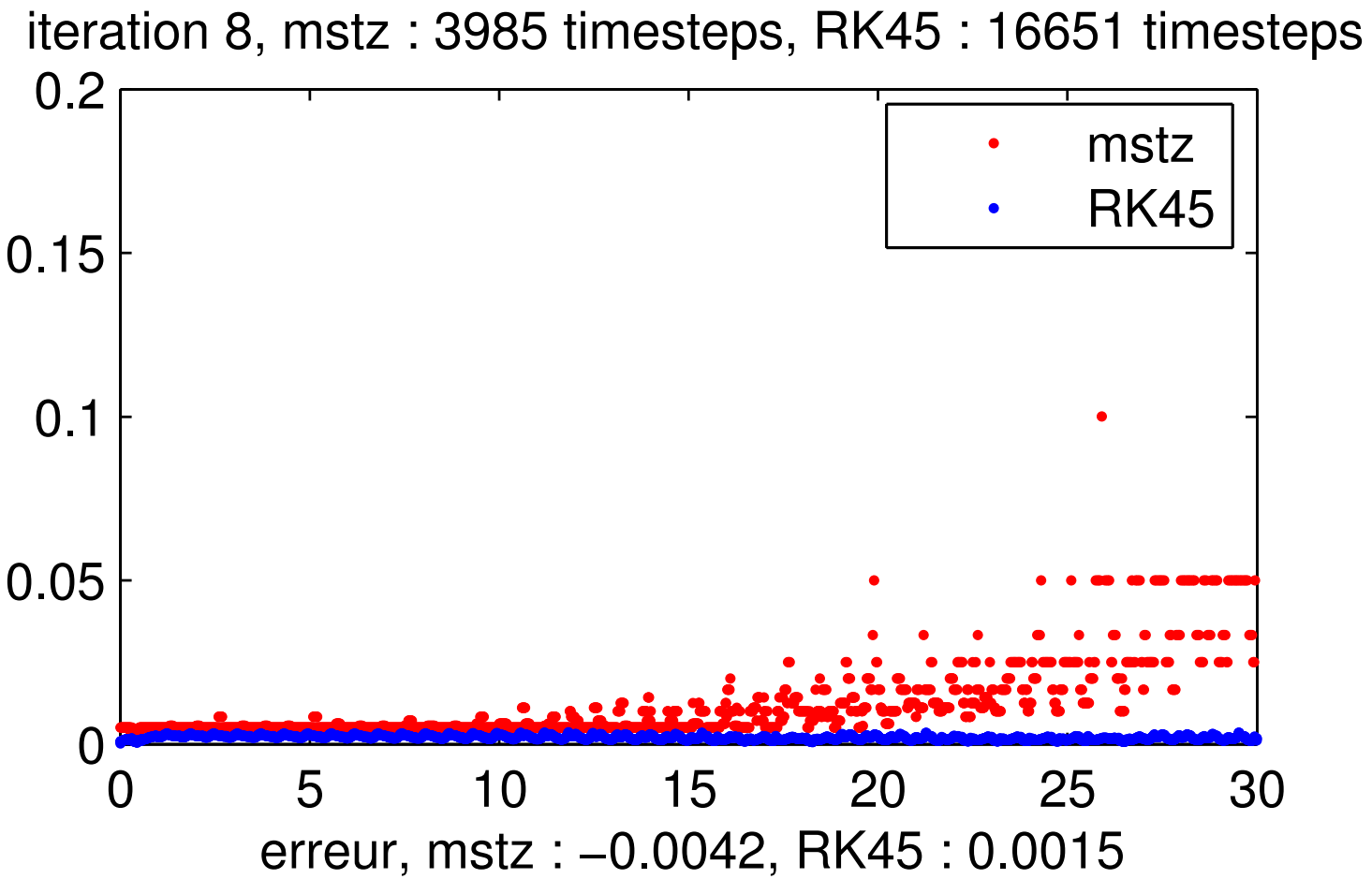
Size of time steps (Lorenz problem, TOL= 0.01)



Size of time steps (Lorenz problem, TOL= 0.01)



Size of time steps (Lorenz problem, TOL= 0.01)



Comparison with Szepessy and al.

	TOL	|True error|	N	N_{tot}
`mstz` demi-pas	0.1	0.02	6352	10524
`mstz` Szepessy	0.1	0.01	6000	20000
`mstz` demi-pas	0.01	0.004	7970	12826
`mstz` Szepessy	0.01	0.003	9000	34000

Application to PDE

We now use the algorithm `mstz` to solve semi-discretization of partial differential equations. We use the heat equation to validate our code.

1. We study the properties of the Burgers Equation (a non-chaotic equation) with the help of `mstz`. We use a centered finite difference scheme for the spatial discretization.
2. We compare `RK45` and `mstz` on the Kuramoto-Sivashinsky Equation, a chaotic PDE. We use the pseudo-spectral method for the spatial discretization.

Application to PDE

We now use the algorithm `mstz` to solve semi-discretization of partial differential equations. We use the heat equation to validate our code.

1. We study the properties of the Burgers Equation (a non-chaotic equation) with the help of `mstz`. We use a centered finite difference scheme for the spatial discretization.
2. We compare `RK45` and `mstz` on the Kuramoto-Sivashinsky Equation, a chaotic PDE. We use the pseudo-spectral method for the spatial discretization.

Application to PDE

We now use the algorithm `mstz` to solve semi-discretization of partial differential equations. We use the heat equation to validate our code.

1. We study the properties of the Burgers Equation (a non-chaotic equation) with the help of `mstz`. We use a centered finite difference scheme for the spatial discretization.
2. We compare `RK45` and `mstz` on the Kuramoto-Sivashinsky Equation, a chaotic PDE. We use the pseudo-spectral method for the spatial discretization.

Study of Burgers Equation

For $x \in [-1\ 1]$ and $t \in [0\ 1]$, we have

$$\begin{cases} \dfrac{\partial}{\partial t}u(t,x) + \dfrac{\partial}{\partial x}\dfrac{u^2(t,x)}{2} = 0 \\ u(0,x) = u_0(x) \end{cases}$$

Study of Burgers Equation

For $x \in [-1\ 1]$ and $t \in [0\ 1]$, we have

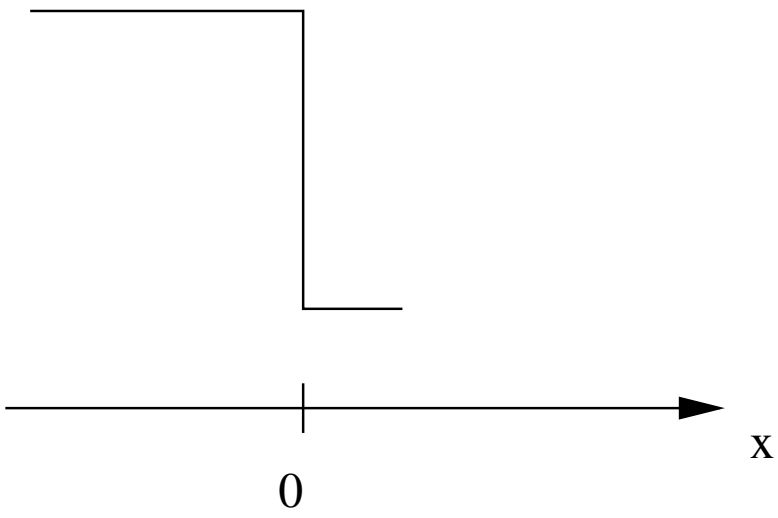
$$\begin{cases} \dfrac{\partial}{\partial t}u(t,x) + \dfrac{\partial}{\partial x}\dfrac{u^2(t,x)}{2} = \varepsilon \dfrac{\partial^2}{\partial x^2}u(t,x) \\ u(0,x) = u_0(x) \end{cases}$$

Viscous term

Centered Finite Difference Scheme

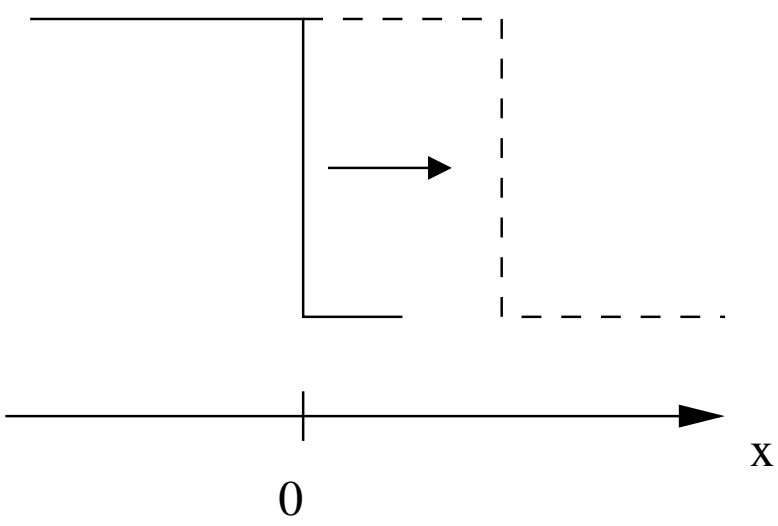
$$\dot{u}_j^h(t) = -\frac{\left(u_{j+1}^h(t)\right)^2 - \left(u_{j-1}^h(t)\right)^2}{4h} + \varepsilon \frac{u_{j+1}^h(t) - 2u_j^h(t) + u_{j-1}^h(t)}{h^2}$$

Initial conditions



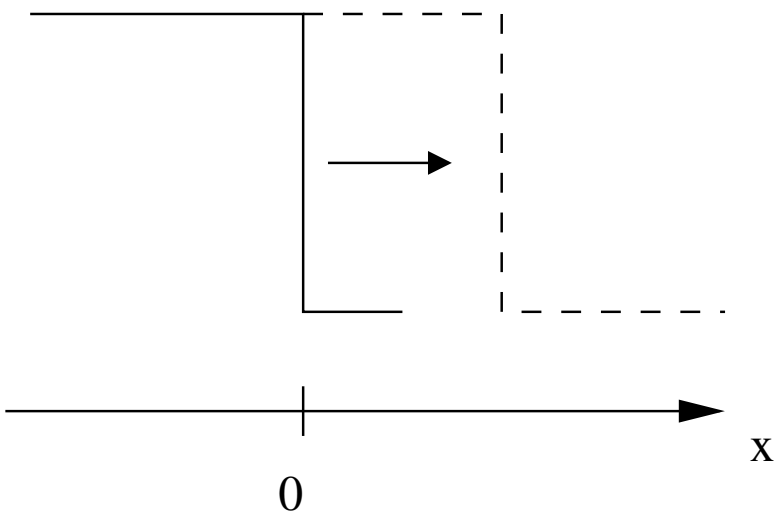
Shock Wave

Initial conditions

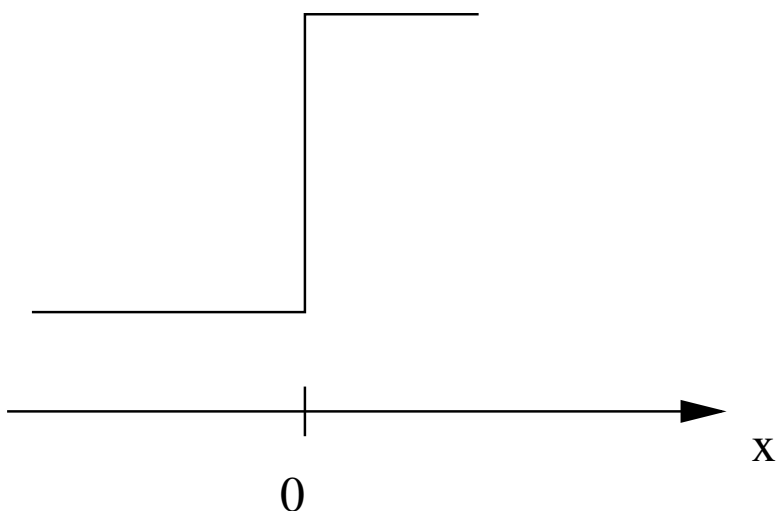


Shock Wave

Initial conditions

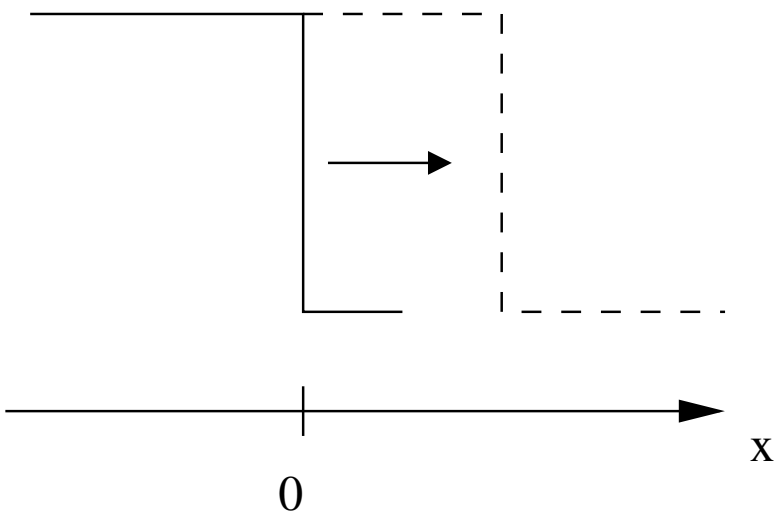


Shock Wave

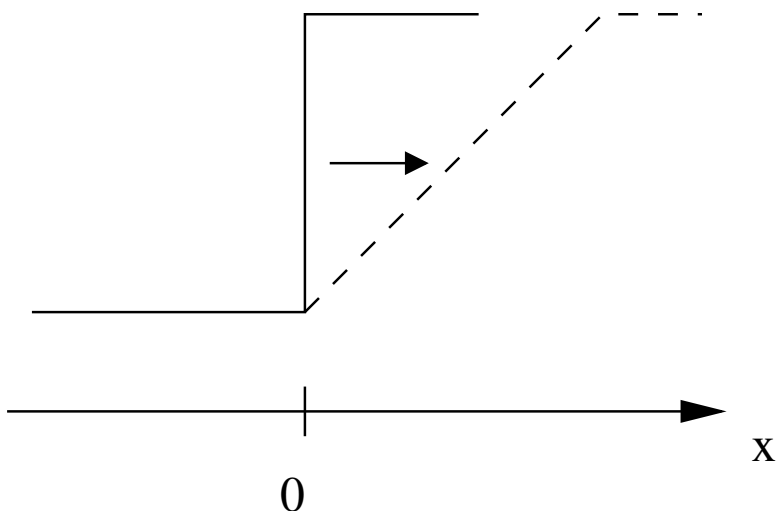


Rarefaction Wave

Initial conditions

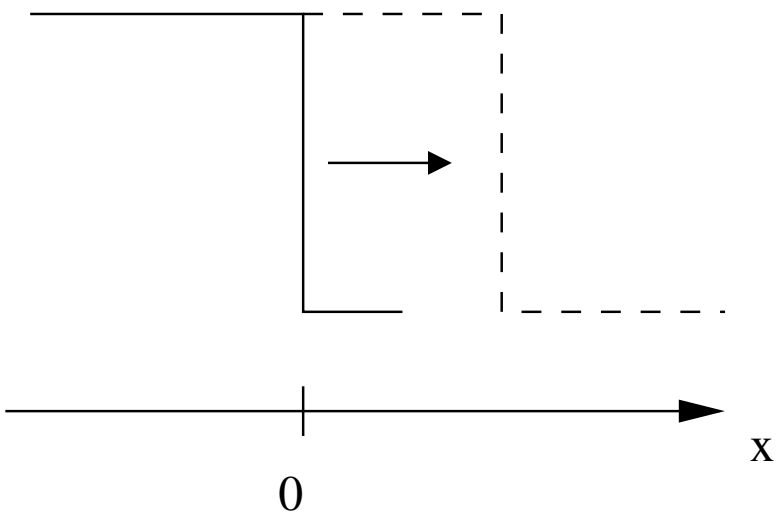


Shock Wave

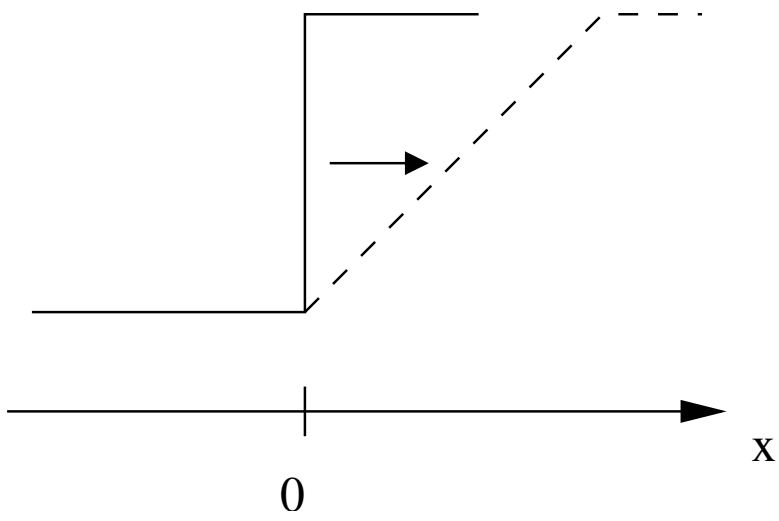


Rarefaction Wave

Initial conditions



Shock Wave



Rarefaction Wave

Which one is the hardest to solve ?

Goal Function

For g, we select a point $\bar{x} \in [-1\ 1]$. Our goal is to have a very precise value of $u(T, \bar{x})$. To have a smoother function in the x-domain, we take a weighted sum around the point $\bar{x}$. The weights are give by the normal density.

$$g(u(T)) = \sum_{j=0}^{M-1} w_j u(T, x_j) = \sum_{j=0}^{M-1} \frac{1}{\sqrt{2\pi}\sigma} \mathrm{e}^{-\frac{1}{2}\left(\frac{x_j - \bar{x}}{\sigma}\right)^2} u(T, x_j)$$

$$(\nabla g(u(T)))_j = w_j = \frac{1}{\sqrt{2\pi}\sigma} \mathrm{e}^{-\frac{1}{2}\left(\frac{x_j - \bar{x}}{\sigma}\right)^2}$$

Parameters

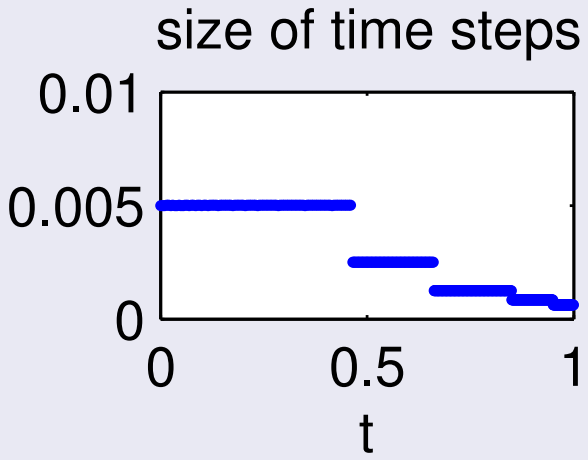
- ε=0.01
- h=0.025
- Δt=0.005 (initial)
- $\bar{x} = 0.5$, $\sigma = 0.1$.
- TOL=0.005
- Numerical method M : Euler progressive

Shock Wave

Rarefaction Wave

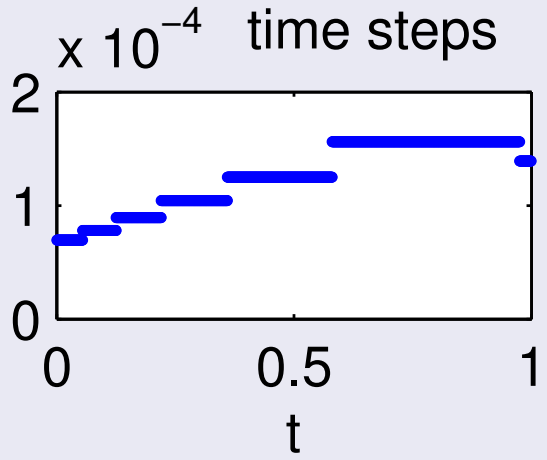
Comparison

Shock Wave (down)



- Number of timesteps needed : 523
- Number of evaluations needed : 3168

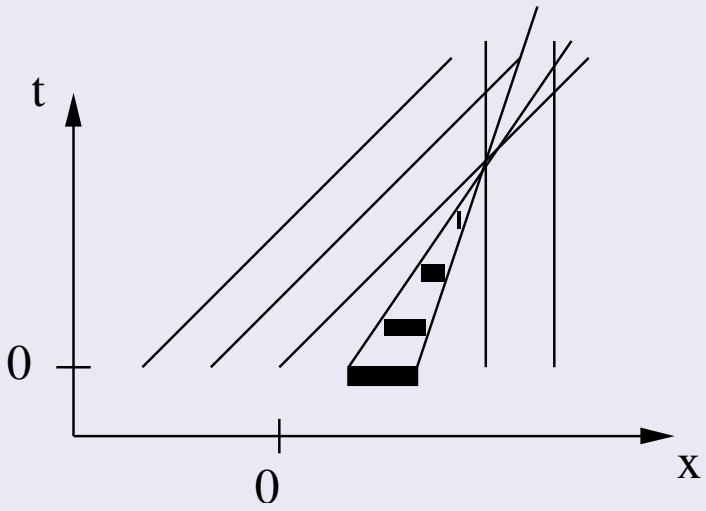
Rarefaction Wave (up)



- Number of timesteps needed : 8565
- Number of evaluations needed : 48750

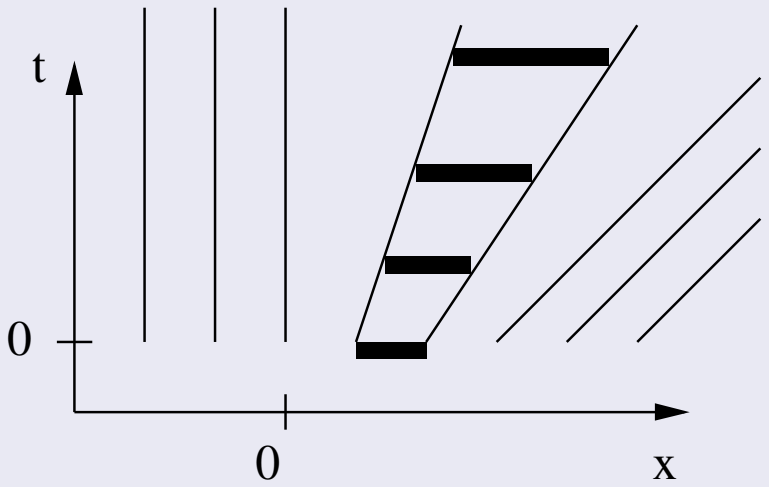
Explanation

Shock Wave (down)



- The perturbations travel quicker than the wave. They are eaten by the shock
- The important parts are the parts just prior T

Rarefaction Wave (up)



- The perturbations are amplified along the time interval
- It's important to make good calculations at the begining

A chaotic PDE

The Kuramoto-Sivashinsky Equation. $x \in \mathbb{R}$, $t \in [0\ 120]$.

$$\begin{cases} \dfrac{\partial u(t,x)}{\partial t} &= -\dfrac{\partial^2 u(t,x)}{\partial x^2} - \dfrac{\partial^4 u(t,x)}{\partial x^4} - \dfrac{1}{2}\dfrac{\partial u^2(t,x)}{\partial x} \\ u(0,x) &= u_0(x) \\ u(t,x) &= u(t,x+l) \end{cases}$$

A chaotic PDE

The Kuramoto-Sivashinsky Equation. $x \in \mathbb{R}$, $t \in [0\ 120]$.

$$\begin{cases} \dfrac{\partial u(t,x)}{\partial t} &= -\dfrac{\partial^2 u(t,x)}{\partial x^2} - \dfrac{\partial^4 u(t,x)}{\partial x^4} - \dfrac{1}{2}\dfrac{\partial u^2(t,x)}{\partial x} \\ u(0,x) &= u_0(x) \\ u(t,x) &= u(t,x+l) \end{cases}$$

Inverse Heat term. Amplifies the perturbations

A chaotic PDE

The Kuramoto-Sivashinsky Equation. $x \in \mathbb{R}$, $t \in [0\ 120]$.

$$\begin{cases} \dfrac{\partial u(t,x)}{\partial t} &= -\dfrac{\partial^2 u(t,x)}{\partial x^2} - \dfrac{\partial^4 u(t,x)}{\partial x^4} - \dfrac{1}{2}\dfrac{\partial u^2(t,x)}{\partial x} \\ u(0,x) &= u_0(x) \\ u(t,x) &= u(t,x+l) \end{cases}$$

Beam term. Reduces the perturbations

A chaotic PDE

The Kuramoto-Sivashinsky Equation. $x \in \mathbb{R}$, $t \in [0\ 120]$.

$$\begin{cases} \dfrac{\partial u(t,x)}{\partial t} &= -\dfrac{\partial^2 u(t,x)}{\partial x^2} - \dfrac{\partial^4 u(t,x)}{\partial x^4} - \dfrac{1}{2}\dfrac{\partial u^2(t,x)}{\partial x} \\ u(0,x) &= u_0(x) \\ u(t,x) &= u(t,x+l) \end{cases}$$

Burgers term. Non-linear transport term

A chaotic PDE

The Kuramoto-Sivashinsky Equation. $x \in \mathbb{R}$, $t \in [0\ 120]$.

$$\begin{cases} \dfrac{\partial u(t,x)}{\partial t} &= -\dfrac{\partial^2 u(t,x)}{\partial x^2} - \dfrac{\partial^4 u(t,x)}{\partial x^4} - \dfrac{1}{2}\dfrac{\partial u^2(t,x)}{\partial x} \\ u(0,x) &= u_0(x) \\ u(t,x) &= u(t,x+l) \end{cases}$$

All together : the solution remain bounded !

Pseudo-spectral method

The 4th order term is to heavy to discretize using finite differences. We use the pseudo-spectral method based on Fourier theory. This allows as well an easy way of dealing with periodic boundary conditions. We write :

$$\begin{aligned} u(t,x) &= \sum_{n=-\infty}^{+\infty} c_n(t) \mathrm{e}^{2i\pi n \frac{x}{l}} \\ \text{with } c_n(t) &= \frac{1}{l} \int_0^l u(t,x) \mathrm{e}^{-2i\pi n \frac{x}{l}} dx \end{aligned}$$

Taking a finite number of Fourier coefficients and computing them with the Fast Fourier Transform gives the pseudo-spectral scheme.

The semi-discretization

We obtain the following ODE ($q = \frac{2\pi}{l}$):

$$\dot{c}_n(t) = (n^2q^2 - n^4q^4)c_n(t) - \frac{1}{2}inqd_n(t) \quad \forall n = -m, \ldots, m$$

where $d_n(t)$ are the Fourier coefficients of $u^2(t,x)$. We compute them using the inverse Fourier transform of all the c_n, then squaring all the components and then taking the Fourier transform again.

The semi-discretization

We obtain the following ODE ($q = \frac{2\pi}{l}$):

$$\dot{c}_n(t) = (n^2q^2 - n^4q^4)c_n(t) - \frac{1}{2}inqd_n(t) \quad \forall n = -m, \ldots, m$$

where $d_n(t)$ are the Fourier coefficients of $u^2(t,x)$. We compute them using the inverse Fourier transform of all the c_n, then squaring all the components and then taking the Fourier transform again.

Inverse Heat term. Amplifies the perturbations

The semi-discretization

We obtain the following ODE ($q = \frac{2\pi}{l}$):

$$\dot{c}_n(t) = (n^2q^2 - n^4q^4)c_n(t) - \frac{1}{2}inqd_n(t) \quad \forall n = -m, \ldots, m$$

where $d_n(t)$ are the Fourier coefficients of $u^2(t,x)$. We compute them using the inverse Fourier transform of all the c_n, then squaring all the components and then taking the Fourier transform again.

Beam term. Reduces the perturbations

The semi-discretization

We obtain the following ODE ($q = \frac{2\pi}{l}$):

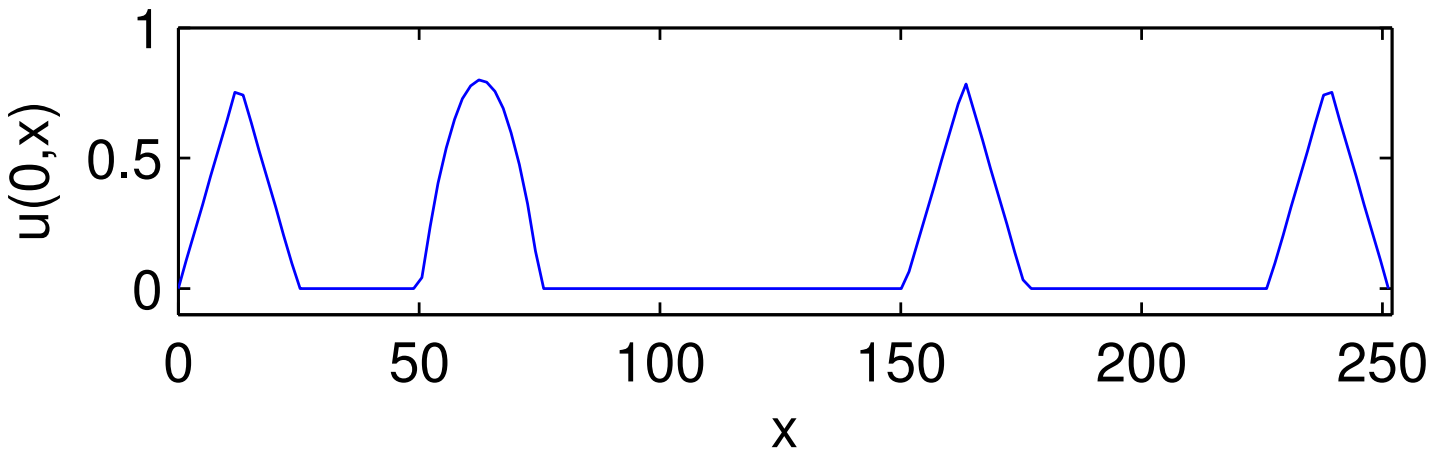
$$\dot{c}_n(t) = (n^2q^2 - n^4q^4)c_n(t) - \frac{1}{2}inqd_n(t) \quad \forall n = -m, \ldots, m$$

where $d_n(t)$ are the Fourier coefficients of $u^2(t,x)$. We compute them using the inverse Fourier transform of all the c_n, then squaring all the components and then taking the Fourier transform again.

Burgers term. Non-linear transport term

Initial condition

Following Wanner and Hairer, the initial condition has the following shape :



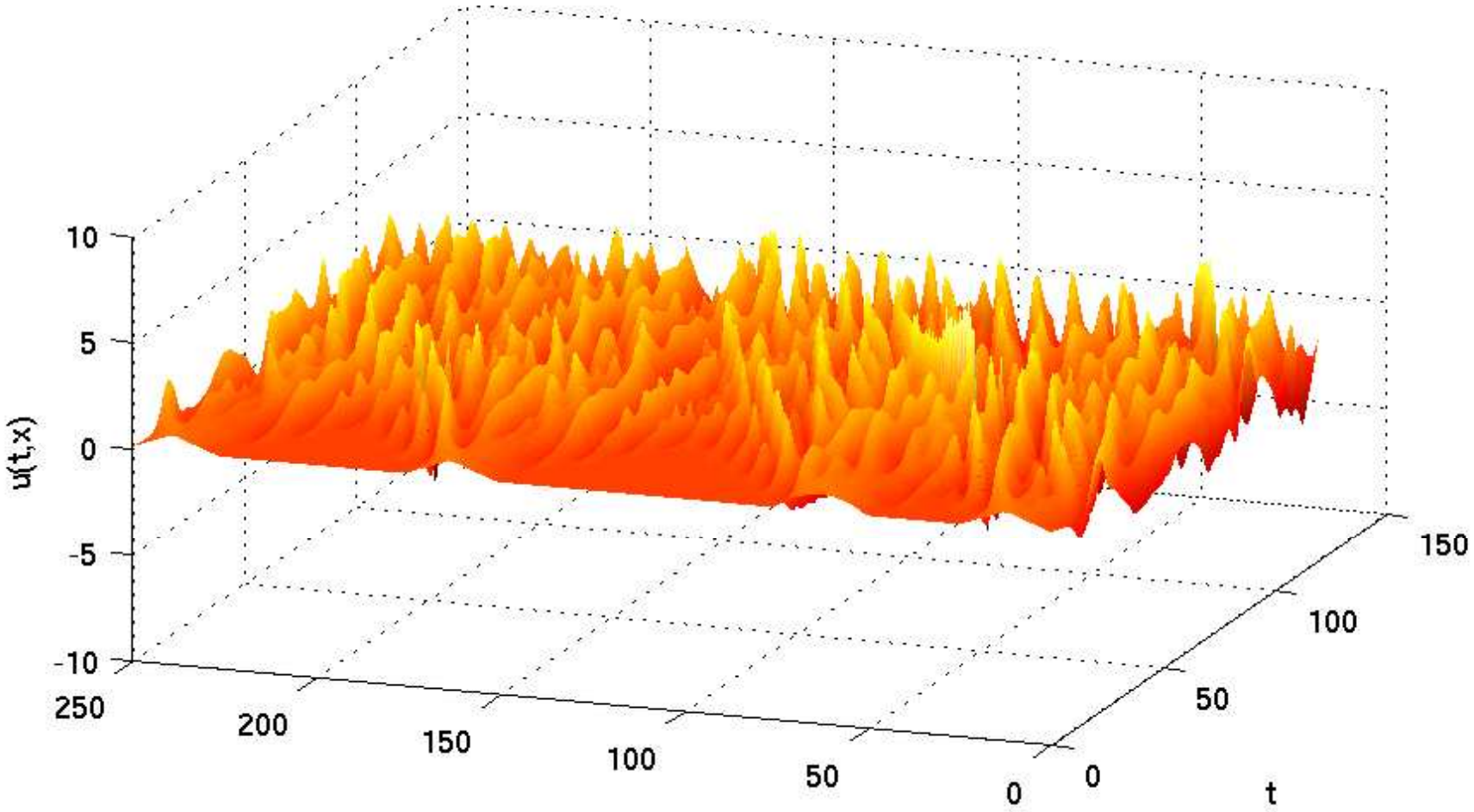
Number of coefficients

To get a solution in a reasonable amount of time, we decided to use 150 Fourier coefficients. The problem is that we can not solve the orginal Kuramoto-Sivashinsky Equation with 150 coefficients. The numerical scheme is unstable. We modify the Kuramoto-Sivashinsky equation as follows :

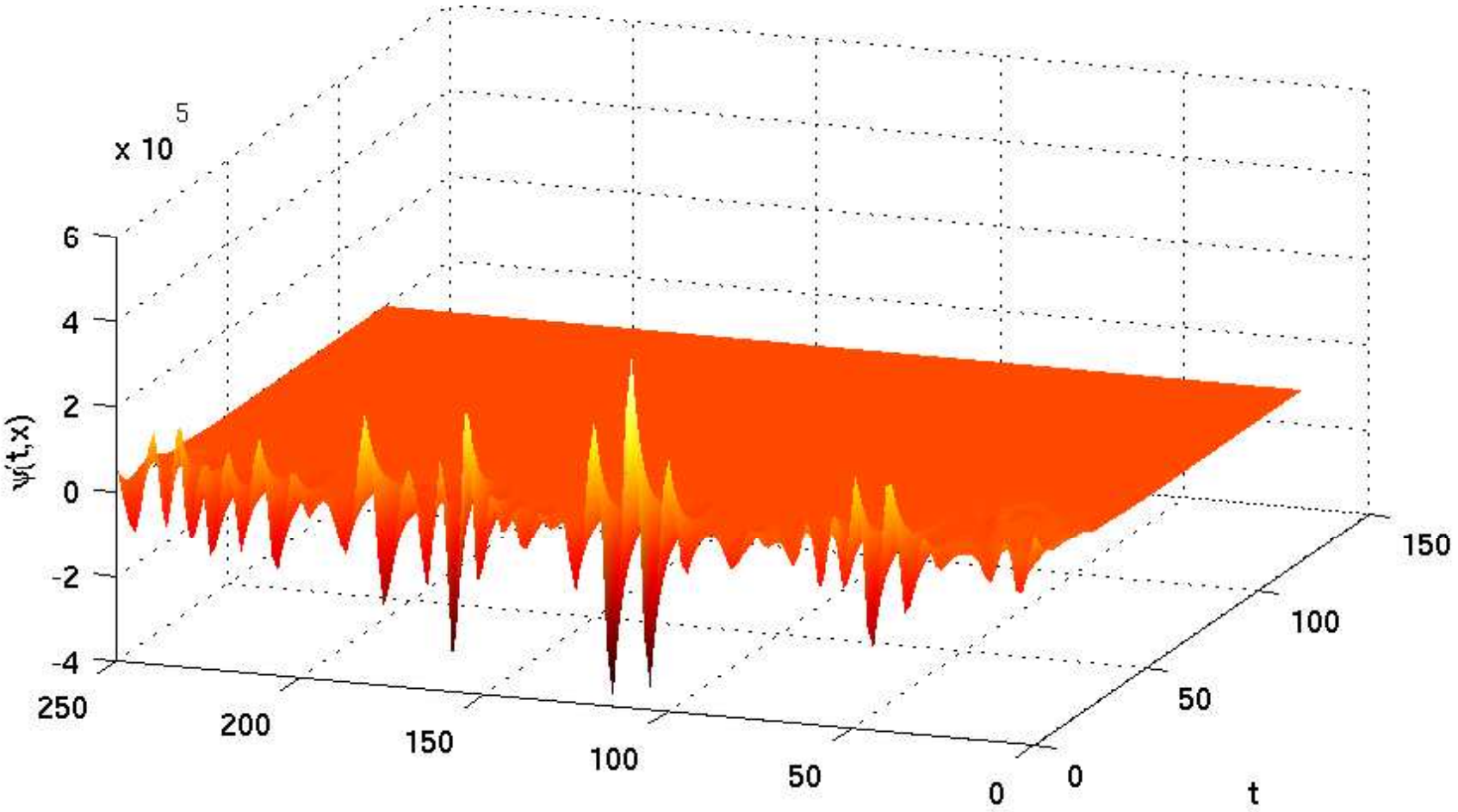
$$\frac{\partial u(t,x)}{\partial t} = -\lambda \frac{\partial^2 u(t,x)}{\partial x^2} - \frac{\partial^4 u(t,x)}{\partial x^4} - \frac{1}{2}\frac{\partial u^2(t,x)}{\partial x}$$

For $\lambda = 1$, we have the original equation. For $\lambda < 1$, the perturbations are less amplified by the inverse heat term, so the equation is easier to solve. We will be able to solve it for $\lambda = 0.93$.

Solution



Solution of the dual



Results

	TOL	$\mathcal{E}^t$	$\overline{\mathcal{E}}^t$	# eval. a	# eval. $(a')^*$	CPU	N	it
`mstz`	0.0001	2.09e-05	3.41e-05	18666	6222	115.94	637	2
`RK45`	0.0001	3.55e-05	-	29499	0	55.49	2382	3
`mstz`	1e-05	-8.01e-06	-8.61e-06	21078	7026	133.53	771	2
`RK45`	1e-05	4.2e-06	-	45915	0	86.2	3726	3
`mstz`	1e-06	4.61e-08	-6.26e-07	25776	8592	158.04	1032	2
`RK45`	1e-06	8.22e-07	-	71853	0	138.16	5855	3
`mstz`	1e-07	4.15e-07	3.24e-09	79254	26418	484.8	2524	3
`RK45`	1e-07			**No solution !**				

Summary

Using an algorithm which monitors the global error instead of one which monitors the local error is really important for chaotic differential equations. It provides :

- An indicator of the accuracy of the solution. With local errors algorithms, we have no garanty that we are close to the true solution even if the local tolerance is really small.
- A better efficiency. We get the solution faster or for a smaller tolerance.
- Computing the solution of the dual less accurately than the solution of the primal gives a really efficient algorithm

Questions