Dissipativity Theory for Algorithm Analysis

Bin Hu Laurent Lessard

University of Wisconsin–Madison

Beyond Convexity, Oaxaca, 2017

Outline

1. **Dissipativity theory** is a framework for analyzing dynamical systems based on the principle of energy conservation.

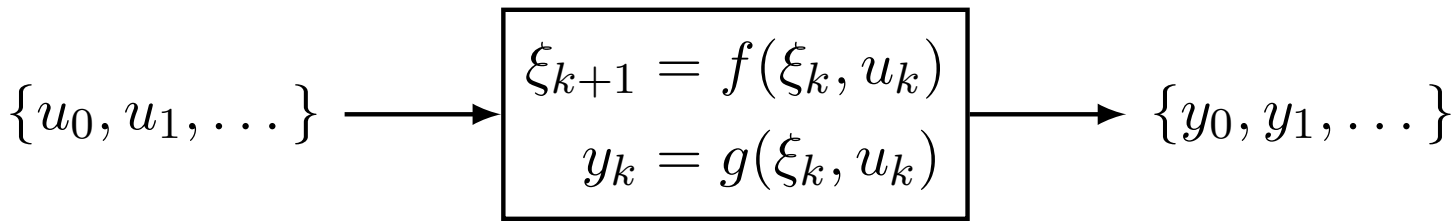
2. **Algorithms are dynamical systems** so we can apply dissipativity theory with an appropriate notion of *energy* and:

   (rate of energy dissipation) $\longleftrightarrow$ (rate of algorithm convergence)

   This gives an intuitive interpretation of algorithm convergence. We'll apply it to Nesterov's method but it's far more general...

Dissipativity theory

We begin with an *open* dynamical system:



A dissipation inequality looks like:

$$\underbrace{V(\xi_{k+1}) - V(\xi_k)}_{\text{increase in stored energy}} \leq \underbrace{S(u_k, y_k)}_{\text{power supplied}}$$

- $V(\xi)$ is called the **storage function**.
- $S(u, y)$ is called the **supply rate**.

Dissipativity theory

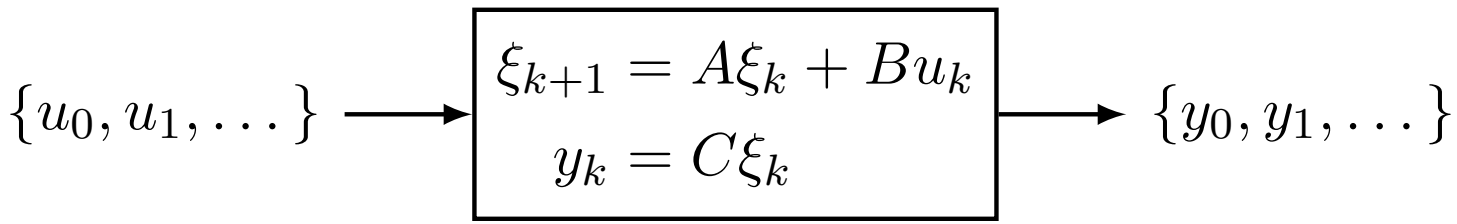
System	**Storage function**	**Supply rate**
Electrical circuit	energy in inductors and capacitors	$V^{\mathsf{T}} I$ (voltage $\times$ current)
Mechanical system	potential + kinetic energy	$F^{\mathsf{T}} v + \tau^{\mathsf{T}} \omega$ (lin. + rot. power)
Thermodynamic system (first law)	internal energy	$Q + W$ (heat + work)
Thermodynamic system (second law)	entropy	$-Q/T$ (heat/temp.)

... now back to optimization algorithms!

Slide credit: J.C. Willems

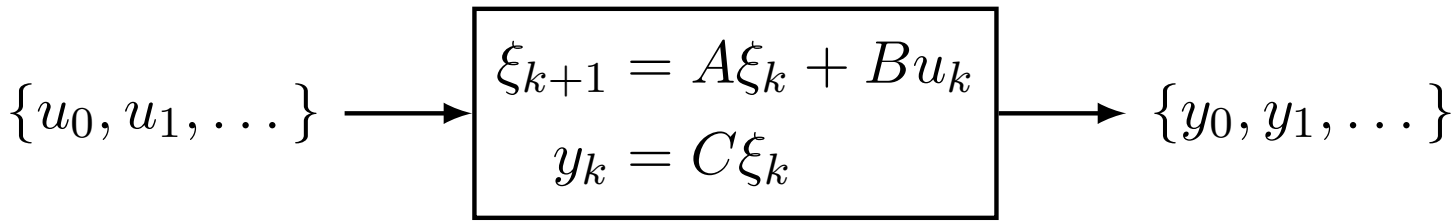
First-order methods

Specialize to a *linear* dynamical system:

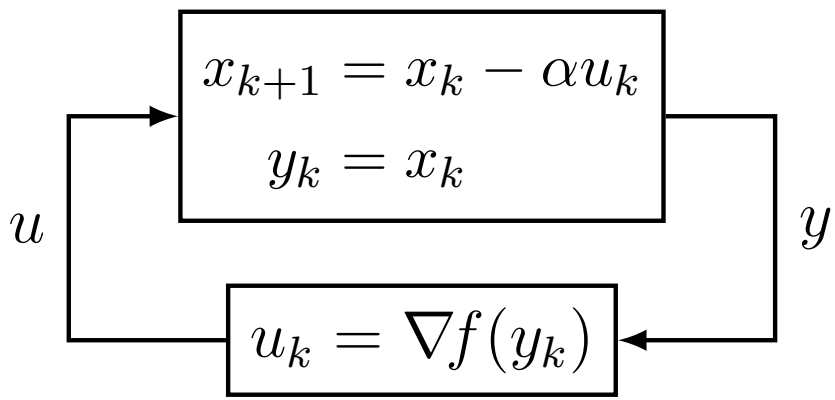


Many first-order methods can be expressed as an open dynamical system with the feedback law $u_k = \nabla f(y_k)$. It's simply a matter of figuring out what A, B, C are.

First-order methods



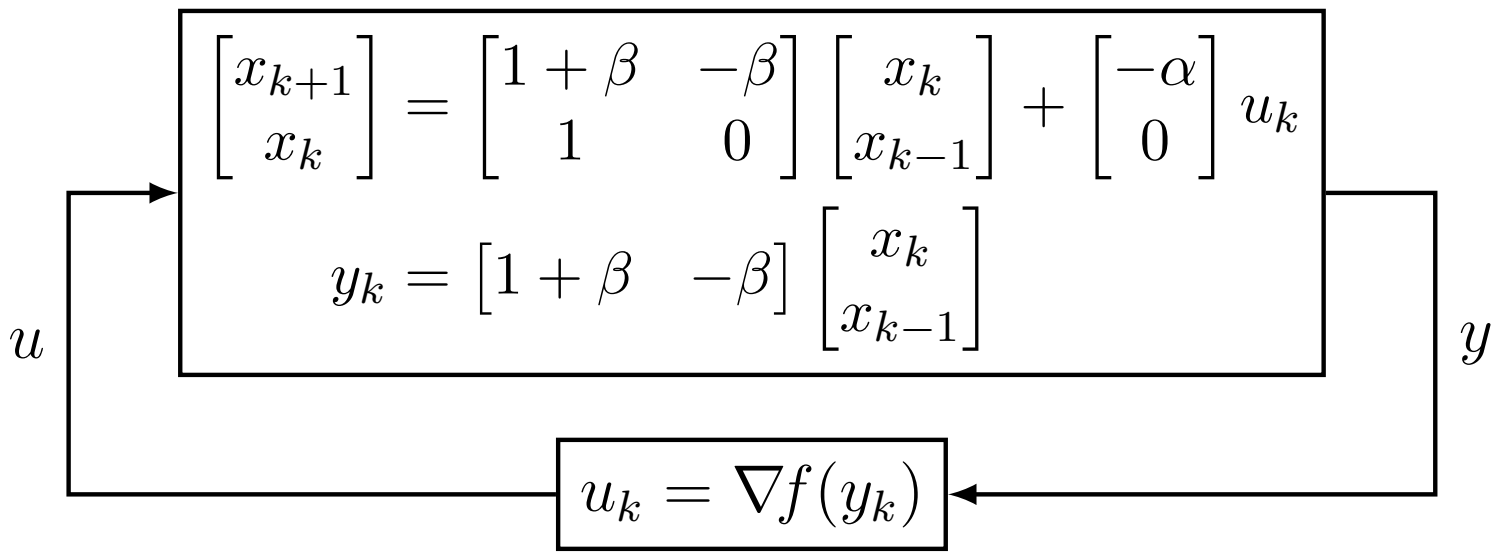
Gradient descent: $x_{k+1} = x_k - \alpha \nabla f(x_k)$.



First-order methods

$$\{u_0, u_1, \dots\} \longrightarrow \boxed{\begin{aligned} \xi_{k+1} &= A\xi_k + Bu_k \\ y_k &= C\xi_k \end{aligned}} \longrightarrow \{y_0, y_1, \dots\}$$

Nesterov's method:
$$\begin{cases} y_k = x_k + \beta(x_k - x_{k-1}) \\ x_{k+1} = y_k - \alpha\nabla f(y_k) \end{cases}$$



From dynamics to dissipation

For algorithms, since $u_k = \nabla f(y_k)$, the supply rate is intimately connected to the function f. We will leverage this fact!

The general recipe is:

1. Properties of f determine the supply rate.
2. Find a storage function that makes a dissipation inequality true when using this supply rate (it's a convex problem).
3. Use the dissipation inequality to obtain a convergence result.

From dynamics to dissipation

For more precise control over convergence rate,
we use the more general dissipation inequality:

$$V(\xi_{k+1}) - \rho^2 V(\xi_k) \leq S(\xi_k, u_k)$$

- parameter $\rho \in [0, 1]$ controls amount of internal dissipation.
- supply rate is allowed to depend explicitly on the state ξ_k.

Example: Gradient descent

with a smooth and strongly convex f

If f is L-smooth and m-strongly convex, then by co-coercivity:

$$\begin{bmatrix} x - x_\star \\ \nabla f(x) \end{bmatrix}^\mathsf{T} \underbrace{\begin{bmatrix} 2mL & -(m+L) \\ -(m+L) & 2 \end{bmatrix}}_{X} \begin{bmatrix} x - x_\star \\ \nabla f(x) \end{bmatrix} \leq 0$$

1. rewrite the dynamics as:

$$\underbrace{(x_{k+1} - x_\star)}_{\xi_{k+1}} = \underbrace{(x_k - x_\star)}_{\xi_k} - \alpha \nabla f(x_k)$$

2. define the supply rate to be:

$$S(\xi, u) := \begin{bmatrix} \xi \\ u \end{bmatrix}^\mathsf{T} X \begin{bmatrix} \xi \\ u \end{bmatrix}$$

Example: Gradient descent

with a smooth and strongly convex f

Whenever $u_k = \nabla f(x_k)$ and $\xi_k = x_k - x_\star$, the supply rate satisfies

$$S(\xi_k, u_k) = \begin{bmatrix} \xi_k \\ u_k \end{bmatrix}^{\mathsf{T}} X \begin{bmatrix} \xi_k \\ u_k \end{bmatrix} \leq 0$$

If we can find a storage function $V(\xi)$ such that

$$V(\xi_{k+1}) - \rho^2 V(\xi_k) \leq S(\xi_k, u_k)$$

If we can find a storage function $V(\xi)$ such that

$$V(\xi_{k+1}) - \rho^2 V(\xi_k) \leq S(\xi_k, u_k)$$

then we conclude that $V(\xi)$ is a Lyapunov function:

$$V(\xi_{k+1}) \leq \rho^2 V(\xi_k)$$

.

Efficiently searching for storage functions

Main technical lemma

Consider the following dynamics and supply rate

$$\xi_{k+1} = A\xi_k + Bu_k \qquad \text{and} \qquad S(\xi, u) := \begin{bmatrix} \xi \\ u \end{bmatrix}^{\mathsf{T}} X \begin{bmatrix} \xi \\ u \end{bmatrix}.$$

If there exists a matrix $P \succeq 0$ such that

$$\begin{bmatrix} A & B \end{bmatrix}^{\mathsf{T}} P \begin{bmatrix} A & B \end{bmatrix} - \rho^2 \begin{bmatrix} I & 0 \end{bmatrix}^{\mathsf{T}} P \begin{bmatrix} I & 0 \end{bmatrix} \preceq X$$

then the dissipation inequality $V(\xi_{k+1}) - \rho^2 V(\xi_k) \leq S(\xi_k, u_k)$ holds with $V(\xi) := \xi^{\mathsf{T}} P \xi$.

Example: Nesterov's accelerated method

with a smooth and strongly convex f

Using the same supply rate as before yields conservative results.
A more intricate supply rate is needed to capture the coupling.

This time, we use relations of the form:

$$\begin{bmatrix}\xi_k\\ u_k\end{bmatrix}^{\mathsf{T}} X_1 \begin{bmatrix}\xi_k\\ u_k\end{bmatrix} \leq f(x_k) - f(x_{k+1}), \quad \begin{bmatrix}\xi_k\\ u_k\end{bmatrix}^{\mathsf{T}} X_2 \begin{bmatrix}\xi_k\\ u_k\end{bmatrix} \leq f(x_\star) - f(x_{k+1})$$

Combining them, we can define:

$$S(\xi_k, u_k) := \begin{bmatrix}\xi_k\\ u_k\end{bmatrix}^{\mathsf{T}} \underbrace{\big(\rho^2 X_1 + (1-\rho^2) X_2\big)}_{X} \begin{bmatrix}\xi_k\\ u_k\end{bmatrix}$$

$$\leq \rho^2\big(f(x_k) - f_\star\big) - \big(f(x_{k+1}) - f_\star\big)$$

Example: Nesterov's accelerated method

with a smooth and strongly convex f

This supply rate satisfies

$$S(\xi_k, u_k) \leq \rho^2\big(f(x_k) - f_\star\big) - \big(f(x_{k+1}) - f_\star\big)$$

If we can find $V(\xi)$ such that

$$V(\xi_{k+1}) - \rho^2 V(\xi_k) \leq S(\xi_k, u_k)$$

If we can find $V(\xi)$ such that

$$V(\xi_{k+1}) - \rho^2 V(\xi_k) \leq S(\xi_k, u_k)$$

then we conclude that $V(\xi) + f(x) - f_\star$ is a Lyapunov function:

$$\big(V(\xi_{k+1}) + f(x_{k+1}) - f_\star\big) \leq \rho^2\big(V(\xi_k) + f(x_k) - f_\star\big)$$

Note: Can efficiently search for V as we did with gradient descent.

Example: Nesterov's accelerated method

with a smooth and strongly convex f

$$\big(V(\xi_{k+1}) + f(x_k) - f_\star\big) \leq \rho^2\big(V(\xi_k) + f(x_{k-1}) - f_\star\big)$$

- Part of the energy is stored in $V(\xi)$, part is stored in $f(x) - f_\star$. Energy can flow back and forth.
- Only the *total energy* is guaranteed to dissipate.

Results

Results: L-smooth and m-strongly convex

The SDP is small in this case and can be solved analytically.

- for Gradient descent, we recover the exact linear rate $\rho = \max(1 - \alpha m, 1 - \alpha L)$.
- for Nesterov's method, we recover the (asymptotically exact) linear rate $\rho = \sqrt{1 - \sqrt{m/L}}$.

Results: L-smooth and (weakly) convex

Main difference:

- We must adjust supply rates (different X)
- Algorithm parameters α_k and β_k now depend on k.
- We expect sublinear rates, so $\rho = 1$.

The SDP is small but depends on k. Can find analytic solution P_k.
Results match the theory:

- for Gradient descent, we recover a $1/k$ rate.
- for Nesterov's method, we recover a $1/k^2$ rate.

So far...

- Algorithm optimization with dynamical systems (SIOPT'16):
  http://www.laurentlessard.com/public/siopt16_iqcopt.pdf
- Analysis of ADMM using dynamical systems (ICML'15):
  http://www.jmlr.org/proceedings/papers/v37/nishihara15.pdf
- Distributed optimization over graphs (Allerton'17):
  www.laurentlessard.com/public/aller17_distrop_toappear.pdf
- Analysis of SAG, SAGA, SDCA, and Finito (COLT'17):
  http://proceedings.mlr.press/v65/hu17b/hu17b.pdf
- **Energy dissipation approach for convex functions (ICML'17):**
  http://proceedings.mlr.press/v70/hu17a/hu17a.pdf

Beyond convexity

- Weak strong convexity [Necoara et.al.'15]:
  $f(x_\star) \geq f(x) + \nabla f(x)^{\mathsf{T}}(x_\star - x) + \frac{m}{2}\|x - x_\star\|^2$
- Restricted secant inequality [Zhang,Yin'13]:
  $\nabla f(x)^{\mathsf{T}}(x - x_\star) \geq m\|x - x_\star\|^2$
- Error Bound [Luo,Tseng'93]: $m\|x - x_\star\| \leq \|\nabla f(x)\|$
- Polyak–Łojasiewicz ['63]: $\frac{1}{2}\|\nabla f(x)\|^2 \geq m(f(x) - f(x_\star))$
- Quadratic Growth [Anitescu'00]: $f(x) - f(x_\star) \geq \frac{m}{2}\|x - x_\star\|^2$
- Lipschitz continuous: $\|\nabla f(y) - \nabla f(x)\| \leq L\|y - x\|$
- One-sided Lipschitz: $f(y) \leq f(x) + \nabla f(x)^{\mathsf{T}}(y - x) + \frac{L}{2}\|y - x\|^2$
- C1 and C2 properties from Shoham's talk

These inequalities are linear in f and quadratic in $\{x, \nabla f\}$.
All compatible with the energy dissipation approach!

Beyond linearity

Dissipativity theory is routinely used to prove results about systems with *nonlinear* dynamics.

- Quadratic storage function may be insufficient. Can use *SOS polynomials* instead (also SDP representable).
- Nonlinear dynamics cause SDP to be nonlinear. Can approximate using polynomial dynamics (also SDP representable).

Algorithms with nonlinear dynamics:
Quasi-Newton methods, Nonlinear Conjugate Gradient, ...

Summary

1. **Dissipativity theory** is a framework for analyzing dynamical systems based on the principle of energy conservation.

2. **Algorithms are dynamical systems** so we can apply dissipativity theory with an appropriate notion of *energy* and:

   (rate of energy dissipation) $\longleftrightarrow$ (rate of algorithm convergence)

3. **Beyond convexity and linearity:** in principle, these tools can certify the performance of *any* dynamical system.

Thanks!

- This material is based upon work supported by the National Science Foundation under Grant No. 1656951.