Parameter Continuation with COCO

Lecture given during the
2016 NZMRI Summer School on
Continuation Methods in Dynamical Systems

Harry Dankowicz

Department of Mechanical Science and Engineering
University of Illinois at Urbana-Champaign

January 11, 2016

Outline

The Catenary Functional

Consider the autonomous two-point boundary-value problem

$$\dot{x}_1 = x_2,\ \dot{x}_2 = (1 + x_2^2)/x_1,\ x_1(0) = 1,\ x_1(1) = p$$

in terms of the vector of state variables $x = (x_1, x_2) \in \mathbb{R}^2$ and the scalar problem parameter $p \in \mathbb{R}$. Solutions correspond to extremal curves $s \mapsto f(s)$, and their derivatives, for the integral functional

$$\int_0^1 f(s)\sqrt{1 + f'(s)^2}\,\mathrm{d}s$$

in the space of functions that satisfy the boundary conditions $f(0) = 1$ and $f(1) = p$.

The Catenary Functional

For arbitrary initial conditions $x_1(0)$ and $x_2(0)$, solutions to the associated initial-value problem are given by

$$x_1(t) = \frac{x_1(0)}{\sqrt{1 + x_2^2(0)}} \cosh\left(\frac{\sqrt{1 + x_2^2(0)}}{x_1(0)} t + \operatorname{arcsinh} x_2(0)\right)$$

and

$$x_2(t) = \sinh\left(\frac{\sqrt{1 + x_2^2(0)}}{x_1(0)} t + \operatorname{arcsinh} x_2(0)\right).$$

The shape is that of a *catenary curve*.

The Catenary Functional

Recall that $x_1(0) = 1$ and $x_1(1) = p$.

For each p, a solution to the boundary-value problem then corresponds to a solution of the nonlinear equation

$$\frac{1}{\sqrt{1 + x_2^2(0)}} \cosh\left(\sqrt{1 + x_2^2(0)} + \operatorname{arcsinh} x_2(0)\right) = p.$$

Since the left-hand side is convex with a unique global minimum at $x_2(0) \approx -2.26$, it follows that there are no solutions to the boundary-value problem for $p \lesssim 0.587$ and two solutions for $p \gtrsim 0.587$.

The Catenary Functional

The MATLAB-based Computational Continuation Core (COCO) enables an approximate analysis of the catenary boundary-value problem, even without access to the closed-form solution.

We construct a family of approximate solutions for admissible values of p by

1. constructing a family of approximate trajectory segments that satisfy $x_1(0) = 1$ and $x_2(0) = 0$, but are defined only on the interval $[0, T]$ for $T \in [0, 1]$;
2. constructing a family of approximate trajectory segments on the interval $[0, 1]$ that satisfy $x_1(0) = 1$.

Here, continuation implements the classical method of *shooting*.

The Catenary Functional

Encode the vector field in the anonymous function `cat`, as shown in the following command

```
>> cat = @(x,p) [x(2,:); (1+x(2,:).^2)./x(1,:)];
```

The encoding is vectorized and autonomous.

A corresponding trajectory segment is given by the single-point time history assigned below to the `t0` and `x0` variables.

```
>> t0 = 0;
>> x0 = [1 0];
```

Here, `t0` encodes a one-dimensional array of time instances and `x0` encodes a two-dimensional array of the corresponding points in state space, with one row per time instant.

The Catenary Functional

We compute a family of trajectory segments under variations in T by invoking the `coco` entry-point function as shown in the sequence of commands below.

```
>> prob = coco_prob();
>> prob = ode_isol2coll(prob, '', cat, t0, x0, []);
>> data = coco_get_func_data(prob, 'coll', 'data');
>> maps = data.coll_seg.maps;
>> prob = coco_add_pars(prob, 'pars', ...
     [ maps.x0_idx; maps.x1_idx(1); maps.T_idx ], ...
     { 'y1s' 'y2s' 'y1e' 'T' });
>> cont_args = { 1, { 'T' 'y1e'  }, [0 1] };
>> coco(prob, 'coll1', [], cont_args{:});
```

Each such segment is an approximate discretization of the exact solution $x_1(t) = \cosh t$, $x_2(t) = \sinh t$. In particular, $x_1(T)$ ranges between $\cosh 0 = 1$ and $\cosh 1 \approx 1.5431$.

The Catenary Functional

We compute a family of trajectory segments under variations in T by invoking the `coco` entry-point function as shown in the sequence of commands below.

```
>> prob = coco_prob();
>> prob = ode_isol2coll(prob, '', cat, t0, x0, []);
>> data = coco_get_func_data(prob, 'coll', 'data');
>> maps = data.coll_seg.maps;
>> prob = coco_add_pars(prob, 'pars', ...
     [ maps.x0_idx; maps.x1_idx(1); maps.T_idx ], ...
     { 'y1s' 'y2s' 'y1e' 'T' });
>> cont_args = { 1, { 'T' 'y1e'  }, [0 1] };
>> coco(prob, 'coll1', [], cont_args{:});
```

Each such segment is an approximate discretization of the exact solution $x_1(t) = \cosh t$, $x_2(t) = \sinh t$. In particular, $x_1(T)$ ranges between $\cosh 0 = 1$ and $\cosh 1 \approx 1.5431$.

The Catenary Functional

We compute a family of trajectory segments under variations in T by invoking the `coco` entry-point function as shown in the sequence of commands below.

```
>> prob = coco_prob();
>> prob = ode_isol2coll(prob, '', cat, t0, x0, []);
>> data = coco_get_func_data(prob, 'coll', 'data');
>> maps = data.coll_seg.maps;
>> prob = coco_add_pars(prob, 'pars', ...
     [ maps.x0_idx; maps.x1_idx(1); maps.T_idx ], ...
     { 'y1s' 'y2s' 'y1e' 'T' });
>> cont_args = { 1, { 'T' 'y1e'  }, [0 1] };
>> coco(prob, 'coll1', [], cont_args{:});
```

Each such segment is an approximate discretization of the exact solution $x_1(t) = \cosh t$, $x_2(t) = \sinh t$. In particular, $x_1(T)$ ranges between $\cosh 0 = 1$ and $\cosh 1 \approx 1.5431$.

The Catenary Functional

We compute a family of trajectory segments under variations in T by invoking the `coco` entry-point function as shown in the sequence of commands below.

```
>> prob = coco_prob();
>> prob = ode_isol2coll(prob, '', cat, t0, x0, []);
>> data = coco_get_func_data(prob, 'coll', 'data');
>> maps = data.coll_seg.maps;
>> prob = coco_add_pars(prob, 'pars', ...
     [ maps.x0_idx; maps.x1_idx(1); maps.T_idx ], ...
     { 'y1s' 'y2s' 'y1e' 'T' });
>> cont_args = { 1, { 'T' 'y1e'  }, [0 1] };
>> coco(prob, 'coll1', [], cont_args{:});
```

Each such segment is an approximate discretization of the exact solution $x_1(t) = \cosh t$, $x_2(t) = \sinh t$. In particular, $x_1(T)$ ranges between $\cosh 0 = 1$ and $\cosh 1 \approx 1.5431$.

The Catenary Functional

We compute a family of trajectory segments under variations in T by invoking the `coco` entry-point function as shown in the sequence of commands below.

```
>> prob = coco_prob();
>> prob = ode_isol2coll(prob, '', cat, t0, x0, []);
>> data = coco_get_func_data(prob, 'coll', 'data');
>> maps = data.coll_seg.maps;
>> prob = coco_add_pars(prob, 'pars', ...
     [ maps.x0_idx; maps.x1_idx(1); maps.T_idx ], ...
     { 'y1s' 'y2s' 'y1e' 'T' });
>> cont_args = { 1, { 'T' 'y1e'  }, [0 1] };
>> coco(prob, 'coll1', [], cont_args{:});
```

Each such segment is an approximate discretization of the exact solution $x_1(t) = \cosh t$, $x_2(t) = \sinh t$. In particular, $x_1(T)$ ranges between $\cosh 0 = 1$ and $\cosh 1 \approx 1.5431$.

Principles of Continuation

Suppose that the function

$$\Phi : \mathbb{R}^n \mapsto \mathbb{R}^m,\ n \geq m \geq 1$$

is continuously differentiable. The equation

$$\Phi(u) = 0$$

is a *(continuation) zero problem* in the vector u of unknown *continuation variables*. The components of Φ are *zero functions*.

Continuation is a computational method for successively growing a collection of solutions to a zero problem.

A zero problem is *adaptive* if Φ changes during continuation and *non-adaptive* otherwise.

Principles of Continuation

The *dimensional deficit* of a zero problem is the difference $n - m$ between the number of continuation variables and the number of zero functions. The dimensional deficit remains constant during continuation.

A solution u^* of a zero problem is *regular* if

$$\partial_u \Phi(u^*)$$

has full rank. There exists a locally unique manifold of solutions to a zero problem through a regular solution. The manifold's dimension equals the corresponding dimensional deficit.

Solution manifolds of dimension 1 are called *branches*.

Principles of Continuation

During continuation, the function $\Psi : \mathbb{R}^n \mapsto \mathbb{R}^r$ monitors properties of solutions to the zero problem. The components of Ψ are *monitor functions*.

The *extended continuation problem*

$$F(u, \mu) = 0$$

is defined in terms of the function

$$F : (u, \mu) \mapsto \begin{pmatrix} \Phi(u) \\ \Psi(u) - \mu \end{pmatrix}$$

and a vector of *continuation parameters* $\mu \in \mathbb{R}^r$. Its dimensional deficit equals $n - m$. If u^* is a regular solution of the zero problem, then $(u^*, \Psi(u^*))$ is a regular solution of the extended continuation problem.

Principles of Continuation

Choose $\mathbb{I}$ and $\mathbb{J}$ such that $\mathbb{I} \cup \mathbb{J} = \{1, \ldots, r\}$ and $\mathbb{I} \cap \mathbb{J} = \emptyset$. The restriction

$$G : (u, \mu_{\mathbb{J}}) \mapsto F(u, \mu)\Big|_{\mu_{\mathbb{I}} = \mu_{\mathbb{I}}^*}$$

defines a *restricted continuation problem*

$$G\left(u, \mu_{\mathbb{J}}\right) = 0$$

with dimensional deficit $n - m - |\mathbb{I}|$.

If $\left(u^*, \mu_{\mathbb{J}}^*\right)$ is a regular solution of the restricted continuation problem, then u^* is a regular solution of the *reduced continuation problem*

$$\begin{pmatrix} \Phi(u) \\ \Psi_{\mathbb{I}}(u) - \mu_{\mathbb{I}}^* \end{pmatrix} = 0$$

Principles of Continuation

We *construct* a restricted continuation problem by defining the functions Φ and Ψ, and by choosing the index set $\mathbb{I}$ corresponding to *inactive continuation parameters*.

We *initialize* a restricted continuation problem by assigning $\mu_{\mathbb{I}}^*$ and providing an initial solution guess for $(u, \mu_{\mathbb{J}})$ with the expectation that there exists a regular solution $\left(u^*, \mu_{\mathbb{J}}^*\right)$ nearby.

In practice, if u_0 is an initial solution guess for u, then

$$\mu_{\mathbb{I}}^* = \Psi_{\mathbb{I}}(u_0)$$

and $\Psi_{\mathbb{J}}(u_0)$ is an initial solution guess for $\mu_{\mathbb{J}}$.

Principles of Continuation

In an adaptive continuation problem, the number and meaning of the zero functions and the continuation variables may change during continuation.

In contrast, the number and meaning of the monitor functions must remain unchanged also for adaptive continuation problems. They must be encoded accordingly.

We may explore different submanifolds of the solution manifold of the original zero problem by reassigning elements between $\mathbb{I}$ and $\mathbb{J}$.

A reassignment from $\mathbb{J}$ to $\mathbb{I}$ imposes a *constraint* on the solutions to the zero problem. A reassignment from $\mathbb{I}$ to $\mathbb{J}$ *releases* the corresponding continuation parameter.

The Collocation Continuation Problem

For continuation of approximate solutions of the dynamical system

$$\dot{x} = F(t, x, p),\ (x, p) \in \mathbb{R}^n \times \mathbb{R}^q,\ t \in [T_0, T_0 + T],$$

define

$$\Phi : (\upsilon, T_0, T, p) \mapsto \begin{pmatrix} \frac{T}{2N}\ \mathfrak{vec}\left(\kappa_F * F_{cn}\right) - W' \cdot \upsilon \\ Q \cdot \upsilon_{bp} \end{pmatrix}$$

in terms of the column matrix υ of unknown values of the state variables on a mesh of $N(m+1)$ *base points*, and the values

$$F_{cn} = F\left(T0 + Tt_{cn}, \mathfrak{vec}_n\left(W \cdot \upsilon\right), 1_{1,Nm} \otimes p\right)$$

of the vector field evaluated on a set t_{cn} of *Nm collocation nodes* on the interval $[0, 1]$.

The Collocation Continuation Problem

The dimensional deficit of the collocation zero problem equals

$$n + q + 2.$$

For an autonomous vector field, append the monitor function

$$u \mapsto T_0$$

and assign the index of the corresponding continuation parameter to $\mathbb{I}$. The dimensional deficit of the corresponding restricted continuation problem then equals

$$n + q + 1.$$

The Catenary Functional

In COCO, we construct an empty continuation problem using the `coco_prob` command:

```
>> prob = coco_prob();
```

The commands

```
>> cat = @(x,p) [x(2,:); (1+x(2,:).^2)./x(1,:)];
>> t0 = 0;
>> x0 = [1 0];
>> prob = ode_isol2coll(prob, '', cat, t0, x0, []);
```

append the collocation zero problem on a default mesh consisting of 10 intervals with 5 base points and 4 collocation nodes in each interval, associate T_0 with the inactive continuation parameter `'coll.T0'`, and initialize the continuation problem with

$$\upsilon = \mathfrak{vec}\left(1_{1,50} \otimes \begin{pmatrix} 1 & 0 \end{pmatrix}\right),\ T_0 = 0,\ T = 0,\ p = \emptyset.$$

The Catenary Functional

The commands

```
>> data = coco_get_func_data(prob, 'coll', 'data');
>> maps = data.coll_seg.maps;
>> prob = coco_add_pars(prob, 'pars', ...
     [ maps.x0_idx; maps.x1_idx(1); maps.T_idx ], ...
     { 'y1s' 'y2s' 'y1e' 'T' });
```

append the monitor functions

$$u \mapsto \begin{pmatrix} \upsilon_i \\ \upsilon_{f,1} \\ T \end{pmatrix},$$

label the corresponding continuation parameters by `'y1s'`, `'y2s'`, `'y1e'`, and `'T'`, and assign the corresponding indexes to $\mathbb{I}$.

The Catenary Functional

The dimensional deficit of the restricted continuation problem is now -1. The commands

```
>> cont_args = { 1, { 'T' 'y1e'  }, [0 1] };
>> coco(prob, 'coll1', [], cont_args{:});
```

identify the desired manifold dimension as 1, reassign the indexes of the continuation parameters `'T'` and `'y1e'` to $\mathbb{J}$, and restrict continuation to the domain `'T'`$\in [0, 1]$.

By default, the collocation continuation problem is non-adaptive, so the number and meaning of the continuation variables and the zero functions is unchanged during continuation.

The Catenary Functional

As an alternative, the commands

```
>> prob = coco_set(prob, 'cont', 'NAdapt', 10);
>> cont_args = { 1, { 'T' 'y1e'  }, [0 1] };
>> coco(prob, 'coll1', [], cont_args{:});
```

instruct the continuation algorithm to make adaptive changes to the problem discretization after every ten successive steps of continuation.

Such adaptive changes are designed to ensure that a suitably estimated discretization error remains below a critical threshold during continuation.

More frequent changes, or a finer initial mesh, may be required in order to ensure successful continuation across the desired computational domain.

The Bratu Problem

Consider the autonomous two-point boundary-value problem

$$\dot{x}_1 = x_2,\ \dot{x}_2 = -pe^{x_1},\ x_1(0) = 0,\ x_1(1) = 0$$

in terms of the vector of state variables $x = (x_1, x_2) \in \mathbb{R}^2$ and the scalar problem parameter $p \in \mathbb{R}$. When $p = 0$,

$$x_1(t) = x_2(t) = 0$$

is a trivial solution.

We encode the vector field and the boundary conditions in the anonymous functions `brat` and `brat_bc`.

```
>> brat    = @(x,p) [x(2,:); -p(1,:).*exp(x(1,:))];
>> brat_bc = @(~,T,x0,x1,p) [T-1; x0(1); x1(1)];
```

The Bratu Problem

The commands

```
>> coll_args = { brat, [0;1], zeros(2), 0 };
>> bvp_args  = [ coll_args, 'p', { brat_bc } ];
>> prob      = ode_isol2bvp(coco_prob(), bvp_args{:});
```

- construct the collocation zero problem on a default mesh consisting of 10 intervals with 5 base points and 4 collocation nodes in each interval,
- associate p and T_0 with inactive continuation parameters `'p'` and `'coll.T0'`,
- initialize the continuation problem with

$$\upsilon = \mathfrak{vec}\left(1_{1,50} \otimes \begin{pmatrix} 0 & 0 \end{pmatrix}\right),\ T_0 = 0,\ T = 1,\ p = 0,$$

- and append the boundary conditions

$$T = 1,\ \upsilon_{i,1} = \upsilon_{f,1} = 0.$$

The Bratu Problem

The dimensional deficit of the restricted continuation problem now equals 0. The command

```
>> bd = coco(prob, 'brat1', [], 1, 'p', [0 4]);
```

identifies the desired manifold dimension as 1, reassigns the index of the continuation parameter `'p'` to $\mathbb{J}$, and restricts continuation to the domain `'p'`$\in [0, 4]$.

Solutions exist for $p > 0$ provided that

$$p = \frac{4C^2}{1 + \cosh C}$$

for some C. This equation has two roots provided that $0 \leq p < p^* \approx 3.1583$ and no roots for $p > p^*$.

An Invariant Torus

Consider the non-autonomous dynamical system

$$\dot{\rho} = \rho\big(1 + \rho(\cos \omega t - 1)\big), \; \dot{\psi} = \Omega$$

in the polar coordinates ρ and ψ. Then, for $t \gg 1$,

$$\rho(t) \approx \rho^*(t) = \frac{1 + \omega^2}{1 + \omega^2 - \cos \omega t - \omega \sin \omega t}, \; \psi(t) = \Omega t + \psi_0$$

corresponding to motion on an invariant two-dimensional torus $\mathbb{T}$ described by the torus function

$$u : (\theta_1, \theta_2) \mapsto \big(\rho^*(\theta_2/\omega) \cos \theta_1, \rho^*(\theta_2/\omega) \sin \theta_1\big),$$

where $\dot{\theta}_1 = \Omega$ and $\dot{\theta}_2 = \omega$.

An Invariant Torus

The torus dynamics consist of either i) torus-covering trajectories, or ii) a continuous family of periodic orbits.

The definition $v(\varphi, \tau) := u(\varphi + \Omega\tau, \omega\tau)$ implies that

$$v(\varphi, 0) = u(\varphi, 0),\ v(\varphi, 2\pi/\omega) = u(\varphi + 2\pi\Omega/\omega, 0)$$

and

$$\frac{\partial v}{\partial \tau} = F(\tau, v(\varphi, \tau), p).$$

in terms of the original vector field F in cartesian coordinates.

We approximate the torus in terms of a sequence

$$\{2\pi(j-1)/(2M+1)\}_{j=1}^{2M+1}$$

of values of φ, a corresponding discretized approximation of $u(\varphi, 0)$, and a family of admissible trajectory segments $v(\varphi, \tau)$.

An Invariant Torus

Suppose that the boundary conditions take the form

$$\left(\mathcal{F} \otimes I_2\right) \cdot \begin{pmatrix} v\left(\varphi_1, 2\pi/\omega\right) \\ \vdots \\ v\left(\varphi_{2M+1}, 2\pi/\omega\right) \end{pmatrix} = \left(\left(\mathcal{R} \cdot \mathcal{F}\right) \otimes I_2\right) \cdot \begin{pmatrix} v\left(\varphi_1, 0\right) \\ \vdots \\ v\left(\varphi_{2M+1}, 0\right) \end{pmatrix}$$

in terms of the discrete Fourier transform matrix $\mathcal{F}$ and the rotation matrix $\mathcal{R}$ associated with a fixed rotation number Ω/ω.

Moreover, impose the phase condition $v_2(\varphi_1, 0) = 0$ to eliminate degeneracy associated with arbitrary shifts in φ.

An Invariant Torus

In terms of a collocation discretization, the corresponding *all-to-all*, multi-point boundary conditions read

$$\left(\mathcal{F}\otimes I_2\right)\cdot\begin{pmatrix} \upsilon_f^{(1)} \\ \vdots \\ \upsilon_f^{(2M+1)} \end{pmatrix} = \left(\left(\mathcal{R}\cdot\mathcal{F}\right)\otimes I_2\right)\cdot\begin{pmatrix} \upsilon_i^{(1)} \\ \vdots \\ \upsilon_i^{(2M+1)} \end{pmatrix},$$

$$T_0^{(1)} = \cdots = T_0^{(2M+1)} = 0,$$

$$T^{(1)} = \cdots = T^{(2M+1)} = 2\pi/\omega,$$

and

$$\upsilon_{i,2}^{(1)} = 0$$

resulting in a total dimensional deficit equal to 1.

Additional resources

- *Recipes for Continuation*, SIAM, 2013: principles of continuation, vectorization, and collocation.
- *The* 'coll' *toolbox*, SourceForge, 2015: tutorial documentation and demos for production-ready COCO toolbox.
- *Tomorrow's lecture*, NZMRI, 2016: Continuation of single- and multi-segment periodic orbits in smooth and hybrid dynamical systems, coupled problems.