Design and Application of a Two-dimensional Continuation Solver

Bob Melville, Agere Systems
Steve Mackey, Western Michigan University

Major re-write of work described in

`Melville, R. and S. Mackey,`
`A New Algorithm for Two-Dimensional Numerical Continuation,`
`Computers and Math with Applications, Vol. 30, No. 1, pp. 31-46, 1995.`

Three views of numerical continuation

"More unknowns than equations" – A bit more precisely, consider a smooth map

$$f : R^{n+p} \to R^n, \Gamma = \{y \in R^{n+p} \| f(y) = 0 \in R^n\}.$$

1. Implicit definition of a manifold

2. path to a point solution (sometimes called artificial parameter methods)

3. "Response curve/surface" of a physical system

Implicit definition of a manifold

- $\{(x, y) \| x^2 + y^2 - 1 = 0\}$; i.e., one equation in two unknowns. Solution set is a circle (one-dimensional manifold) in the "ambient space" of dimension 2.

- $\{(x, y, z) \| x^2 + y^2 + z^2 - 1 = 0\}$ defines a sphere in three-space;

- zero-set of the mapping $f : R^3 \to R$

$$
\begin{aligned}
&(x^2 + y^2 + z^2 + 2y - 1)[(x^2 + y^2 + z^2 - 2y - 1)^2 - 8z^2] \\
&+16xz(x^2 + y^2 + z^2 - 2y - 1)
\end{aligned}
$$

  gives a Klein bottle!

Artificial parameter continuation – “a means to an end”

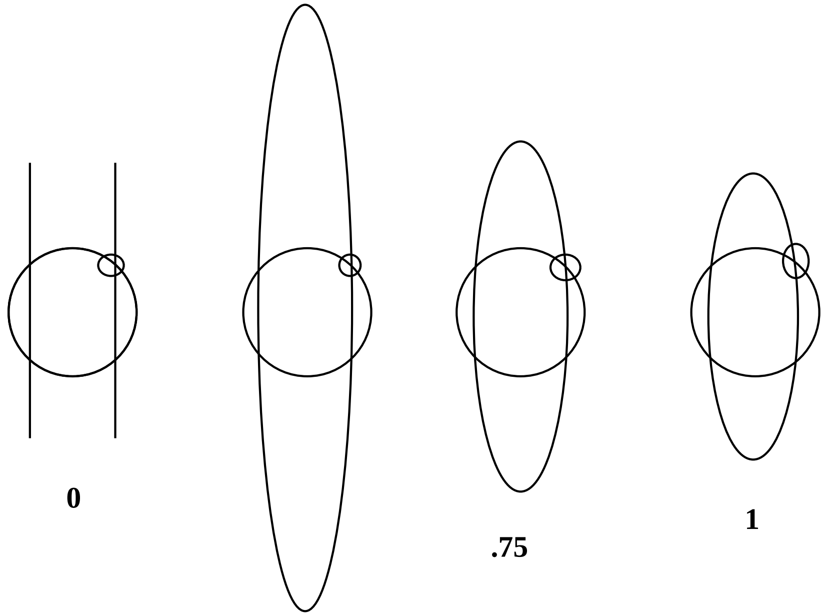
Ultimate goal is to find a zero or root of a smooth mapping $f : R^n \to R^n$; however direct application something like Newton’s (damped or undamped) method can fail:

- diverge to infinity
- cycle
- generate a singular (or ill-conditioned) Jacobian matrix
- converge to a physically unstable solution
- hang on a local minimum (step-size goes to zero)

All these failure modes can be avoided (theoretically and practically) by ”embedding” with an artificial parameter λ.

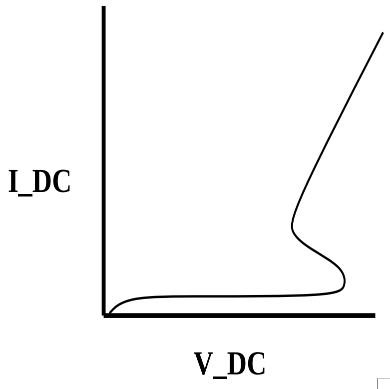
Artificial parameter continuation continued ...

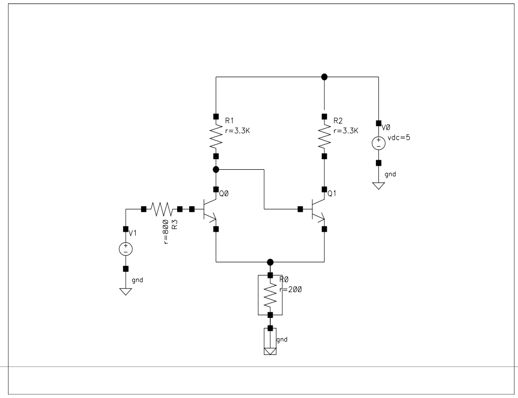
Example – find points of intersection between unit circle and an ellipse $(x^2+y^2-1=0, \frac{x^2}{a}+\frac{y^2}{b}-1=0)$. Modify as $(x^2+y^2-1=0, \frac{x^2}{a}+\lambda\frac{y^2}{b}-1=0)$ and track $\lambda = 0\ldots1$:

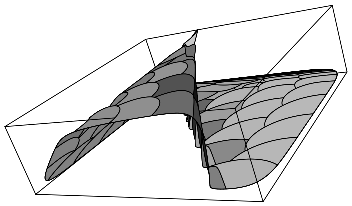


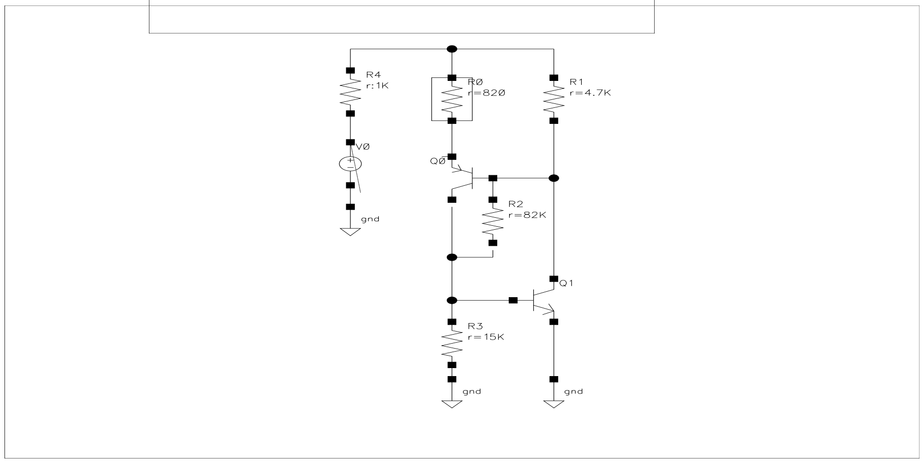
Response curves and surfaces of physical systems

Electronic circuits can exhibit a “snap-back” region (i.e., negative resistance):









Considerations for a two-dimensional response surface

Given a smooth mapping$f : R^{n+2} \to R^n$,define

$$\Gamma = \{y \in R^{n+2} \| f(y_1, \ldots, y_n; y_{n+1}, y_{n+2}) \;=\; 0 \\ (y_{n+1}, y_{n+2}) \in [A_1, B_1] \times [A_2, B_2]\}$$

There are at least three cases to consider:

1. $f(y) = 0$ above has exactly one solution for each point (y_{n+1}, y_{n+2}) in $[A_1, B_1] \times [A_2, B_2]$ (the "exercise region");

2. $f(y) = 0$ has *no* solution for certain points in the exercise region – manifold diverges to infinity (rather common in practical problems);

3. $f(y) = 0$ has more than one solution for certain points in the exercise region – i.e., a "fold".

All three cases can occur in practical problems!

Considerations for a two-dimensional response surface

- Observe that case 1 above can be treated by a simple "stepping" or grid device.

  for $y^*_{n+1} = A_1...B_1$ in steps do
      for $y^*_{n+2} = A_2...B_2$ in steps do
          solve $f(y_1, ..., y_n; y^*_{n+1}, y^*_{n+2}) = 0$ with Newton's method

  Use a previous, nearby grid-point as a starting point for Newton's method. However, even in this case a genuine continuation methods may be more efficient because it uses tangent information.

- Use `implicitplot3d` in `Maple`: Only works for a function $f : R^3 \to R^1$ and requires symbolic expression for function defining manifold – difficult to link into a circuit simulator.

- Genuine multi-dimensional continuation solvers:

  `Rheinboldt, W.,‘On the computation of multi-dimensional solution manifolds of parametrized`

equations, Numer. Math.,53, pp. 165-181, 1988.

Hohman, A., An adaptive continuation method for
implicitly defined surfaces,
SC91-20, Konrad Zuse Zentrum for Infor., Berlin, 1991.

Brodzik, M.L., The Computation of Simplicial Approximations of
Implicitly Defined p-Manifolds. Computers and Mathematics
with Applications, 36(6):93-113, 1998.

Henderson, M.E.,"Multiple Parameter Continuation: Computing Implicitly
Defined k-Manifolds", International Journal of Bifurcation and Chaos,
Vol. 12, No.3 (2002) 451-476

Mike Henderson has an extensive web page!
http://www.research.ibm.com/people/h/henderson/Continuation/
ContinuationMethods.html

The ideal (?) interface

Forward call model: User provides two subroutines:

```
void f(
double* x,    /* [n+2] */
double* rsd   /* [n] */
)
void Df(
double* x,    /* [n+2] */
double* Df,   /* [n \times (n+2)] */
int ldDf
)
```

Reverse call:

```
con2d(x,f,Df,&code);
while( code < 3 ){
  switch( code ){
  case 1: f := residual at x; break;
  case 2: Df := Jacobian matrix at x; break;
  }
  con2d(x,f,Df,&code);
}
```

The software architecture of con2d

Requirements:

- con2d
- one-dimensional continuation solver: e.g., Hompack or Pitcon
- graphical surface rendering tool: e.g., GeomView
- means to evaluate manifold equations and associated Jacobian matrix
- a "seed point" at a corner of the exercise region $f(x_0; A_1, A_2) = 0$

1. achieve re-use;
2. con2d is rather small and simple;
3. immediately incorporate upgrades to underlying one-dimensional solver;
4. extension to co-dimension higher than two is impossible

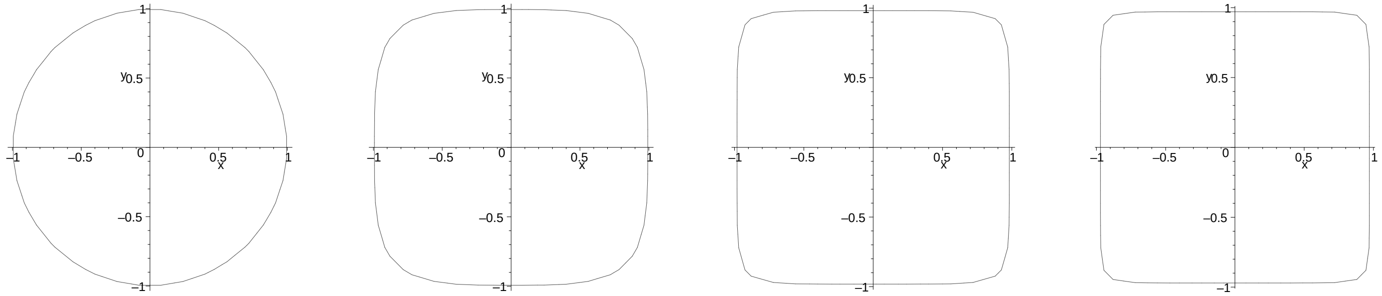
The first step of `con2d` – getting the boundary curve.

User prescribes bounds on two or more coordinates: $A_k \le y_k \le B_k$; Define a smooth approximation to a rectangular bounding box

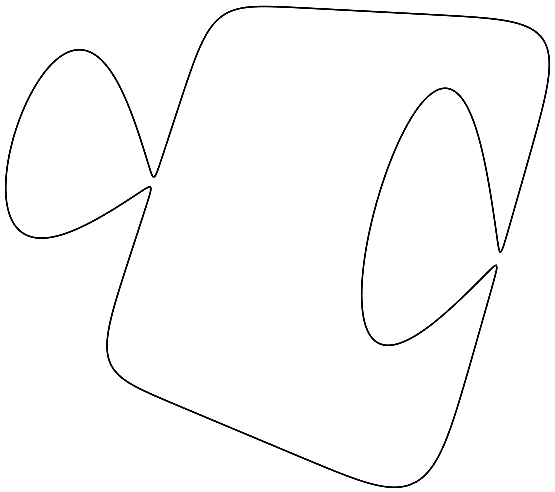
$$K - \sum_{i=1}^{K} \frac{(y_i - \mathtt{mpt}_i)^p}{\mathtt{wid}_i^p} = 0$$

where $\mathtt{mpt}_i = (A_i + B_i)/2$, $\mathtt{wid}_i = (B_i - A_i)/2$, and p is an even integer.

box around $[A_1, B_1] \times [A_2, B_2]$ for different values of p (generated with Maple):



Append above as an additional *augmenting equation* to get $n+1$ equations in $n+2$ unknowns. The solution set of the augmented system is a closed curve in R^{n+2} which can be tracked with your favorite conventional one-dimensional continuation solver (Pitcon or Hompack, among others). Example (challenging) boundary curve for ambient space of dimension 3:



Detecting closure

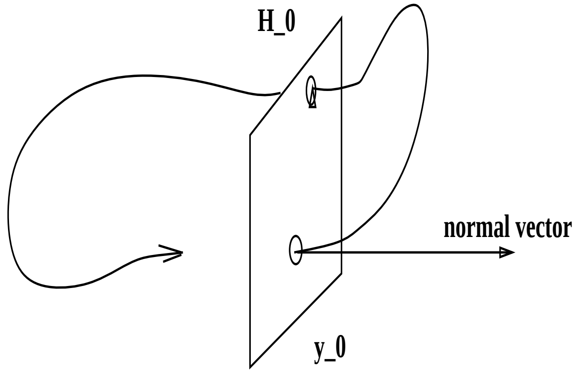
Proximity test:
record seed point y_0;
begin enumerating points on boundary curve $y_1, y_2, ...$;
when an enumerated point y_k comes "close enough" to y_0,declare closure.

At least two problems:

1. obvious start-up problem

2. possibility of false closure

A more sophisticated closure test ...



Tangent vector to boundary curve gives normal of hyperplane H_0 . When two consecutive continuation points straddle H_0, find

$$0 < \sigma < 1, \sigma y_k + (1-\sigma) y_{k+1}$$

nearly on H_0 using a one-dimensional root-finder.

Formulate a system of $n+2$ equations in $n+2$ unknowns:
manifold equations;
equation of boundary curve;
equation of H_0;

closure test continued ...

1. in practice, Newton's method converges in < 5 iterations;

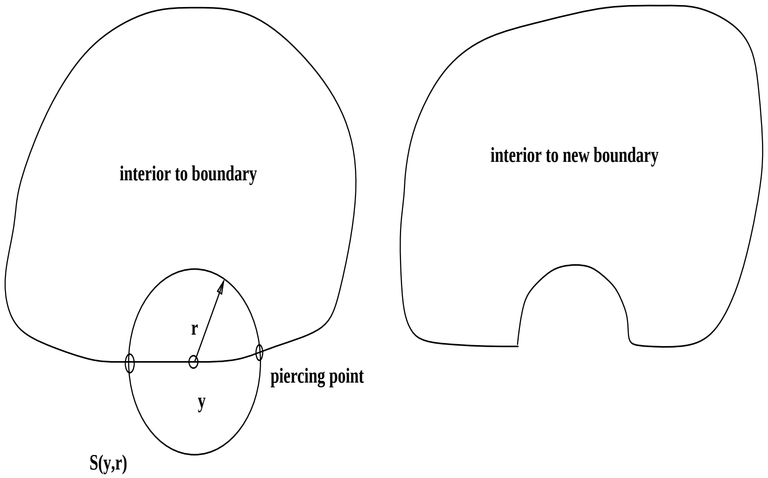
2. declare closure if distance between piercing point and y_0 is smaller than some numerical tolerance which does not depend on the geometry of the manifold;

3. in the case of multiple piercing points, half can be eliminated by a parity test (must cross H_0 going in the proper direction).

4. seems to be reliable in practice, unless step-size of 1d solver is too large

Taking a bite ...

Let y be an arbitrary point on the boundary. Define a hypersphere $S(y,r)$ of radius r centered on y. Intersection of S with manifold has a component which connects the points at which the boundary "exits" S:



1. traverse boundary in both directions from y to find exit points

2. use a piercing computation (manifold,boundary,S) to get exact exit points

3. check that remainder of boundary is exterior to S;

4. if not, then decrease radius of S and try again.

A refinement

Let y be a point on the manifold, $f(y) = 0$; compute an orthogonal basis $\{v_0, v_1\}$ for the tangent plane to the manifold at the point y.

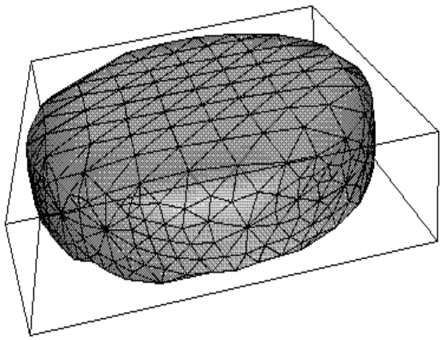
$$\begin{bmatrix} Df@y \\ r_0^T \\ r_1^T \end{bmatrix} v_0 = e_{n+1} + e_{n+2}$$

where r_0, r_1 are **random vectors** of length $n+2$. Normalize v_0.

$$\begin{bmatrix} Df@y \\ v_0^T \\ r_2^T \end{bmatrix} v_1 = e_{n+2}$$

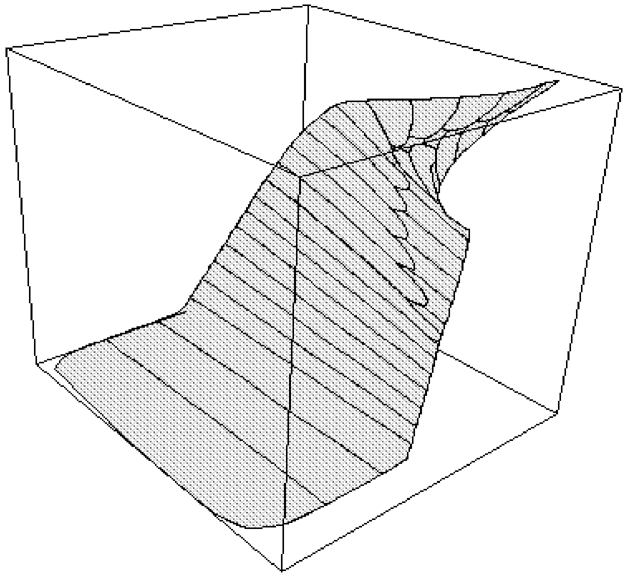
Normalize v_1.

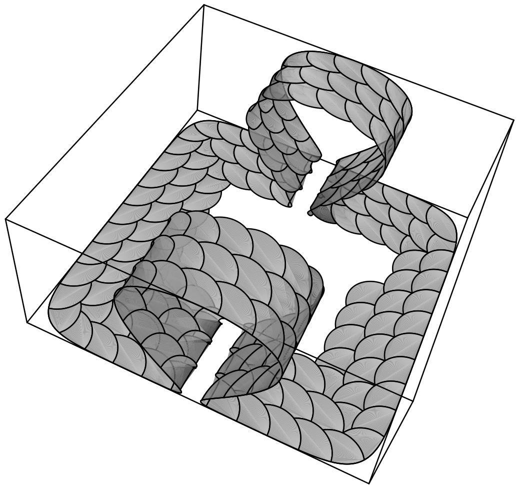
Using tangent plane basis, compute a **flattened sphere** to be used for "bite" computation.

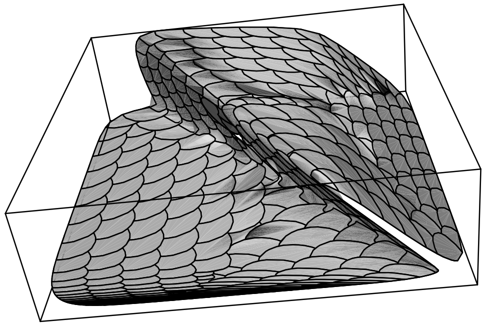


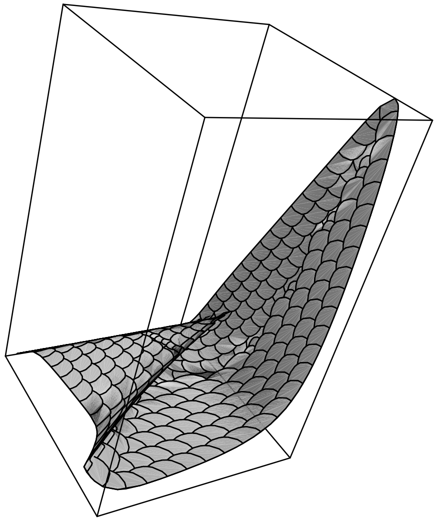
This construction is better able to follow sharp folds in the solution set.

Some pictures ...









Code details

- current code is about 1500 lines of `C` and works with Hompack
- the present version is forward call
- use of dense linear algebra means n less than a hundred or so
- I am willing to try examples that people send me if they provide stand-alone `C` subroutines to evaluate the function and its Jacobian matrix
- depending on interest, I will consider adding a sparse matrix option, reverse call and distributing the code