Ch 8. Solving systems of nonlinear equations, cont'd

For $F: \mathbb{R}^n \to \mathbb{R}^n$ find x^* s.t. $F(x^*) = 0$.

Use Newton to generate $x^{(k)} \underset{\text{hope}}{\to} x^*$, $k = 0, 1, 2, \ldots$

$x^{(k+1)} = x^{(k)} + s^{(k)}$ with $s^{(k)}$ satisfying $F'(x^{(k)}) s^{(k)} = -F(x^{(k)})$.

Let us examine the content and usefulness of some of the theorems in Ch 8.

8.3.6 A Global Convergence Result for Newton's Method

THEOREM 8.8

Let $F : \mathbb{R}^n \to \mathbb{R}^n$ be continuously Fréchet-differentiable and convex over all of $\mathbb{R}^n$. In addition, suppose that $(F'(x))^{-1}$ exists for all $x \in \mathbb{R}^n$ and $(F'(x))^{-1} \geq 0$ for all $x \in \mathbb{R}^n$. Let $F(x) = 0$ have a solution x^*. Then x^* is unique, and the Newton iterates $x^{(k+1)} = x^{(k)} - (F'(x^{(k)}))^{-1} F(x^{(k)})$ converge to x^* for any initial choice $x^{(0)} \in \mathbb{R}^n$. Moreover, for all $k > 0$,

$$x^* \leq x^{(k+1)} \leq x^{(k)} \quad \text{for} \quad k = 1, 2 \ldots . \tag{8.46}$$

(Throughout this theorem statement, the inequalities are interpreted componentwise.)

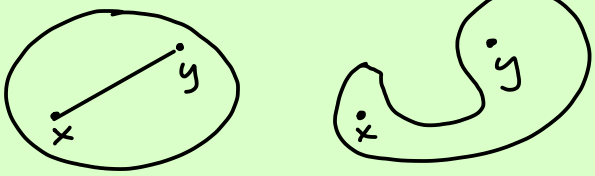
This is rarely useful because the hypotheses are extremely restrictive and rarely met.

A function F is said to be convex on a convex region D if

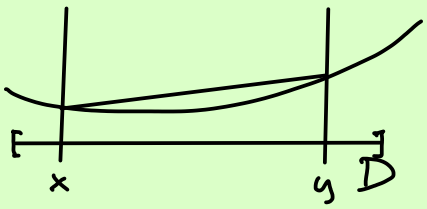
$\forall x, y \in D, \quad (1-\lambda) F(x) + \lambda F(y) \geq F((1-\lambda)x + \lambda y), \quad \lambda \in [0, 1]$

A region D is said to be convex if

$\forall x, y \in D, \quad (1-\lambda)x + \lambda y \in D, \quad \lambda \in [0, 1]$



In 1D (n=1), a convex function is one whose graph is "concave up" in Calc 1 language.



Examples to consider in relation to Thm 8.8

Which if any of the hypotheses do they fail to meet?

If meet all hypotheses, intuitive that Newton is globally convergent to a unique root?

THEOREM 8.8

Let $F : \mathbb{R}^n \to \mathbb{R}^n$ be continuously Fréchet-differentiable and convex over all of $\mathbb{R}^n$. In addition, suppose that $(F'(x))^{-1}$ exists for all $x \in \mathbb{R}^n$ and $(F'(x))^{-1} \geq 0$ for all $x \in \mathbb{R}^n$. Let $F(x) = 0$ have a solution x^*. Then x^* is unique, and the Newton iterates $x^{(k+1)} = x^{(k)} - (F'(x^{(k)}))^{-1}F(x^{(k)})$ converge to x^* for any initial choice $x^{(0)} \in \mathbb{R}^n$. Moreover, for all $k > 0$,

$$x^* \leq x^{(k+1)} \leq x^{(k)} \quad \text{for } k = 1, 2, \ldots. \tag{8.46}$$

(Throughout this theorem statement, the inequalities are interpreted componentwise.)

A: F is F-differentiable everywhere

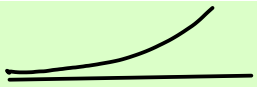
B: F is convex everywhere

C: F' is invertible everywhere

D: F'^-1 >=0 elementwise everywhere

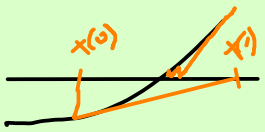
E: F has a root

① $F(x) = e^x$



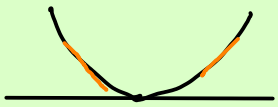
A, B, C (F' nonzero), D, not E DOES NOT APPLY

② $F(x) = e^x - 1$



A, B, C, D, E - DOES APPLY

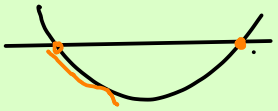
③ $F(x) = x^2$



F not invertible at x=0 (i.e. not C)

not D either - F'^-1 < 0 for x<0

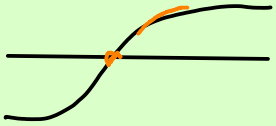
④ $F(x) = x^2 - 1$



A, B, not C, not D, E DOES NOT APPLY

Conclusion clearly false because there are 2 roots

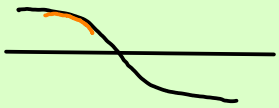
⑤ $F(x) = \tanh x$



A, not B, C, D, E

acutally tanh is a classic example where Newton diverges to infinity for large enough initial x.

⑥ $F(x) = -\tanh x$



Symmetry suggests D could be replaced by non-positivity.

⑦ Example ① from last class

$$F\left(\begin{pmatrix} u \\ v \end{pmatrix}\right) = \begin{bmatrix} u^3 - v \\ u^2 + v^2 - 1 \end{bmatrix}$$

not convex

convex

A, not B, DOES NOT APPLY

Must be, because it has 2 roots,

so not possible to have global convergence to either one.

Convergence of Newton's method

THEOREM 8.6

Assume that $F : D \subset \mathbb{R}^n \to \mathbb{R}^n$ is Fréchet differentiable on an open neighborhood $S_0 \subset D$ of a point $x^* \in D$ for which $F(x^*) = 0$. Also, assume that $F'(x)$ is continuous at x^* and that $F'(x^*)$ is nonsingular. Then x^* is a point of attraction of Newton's method (8.29).

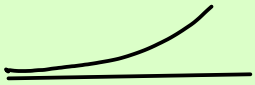
Hypothese are very mild.

i.e. for suff close starting point it converges.

PROOF By Theorem 8.5, with $A(x) = F'(x)$ for $x \in S_0$, we conclude that $G(x) = x - (F'(x))^{-1}F(x)$ is well-defined on some ball $\overline{S}(x^*, \delta) \subset S_0, \delta > 0$. In addition, $\rho(G'(x^*)) = \sigma = 0$. Therefore, by Corollary 8.1, x^* is a point of attraction. □

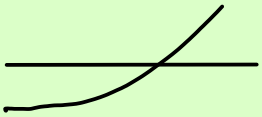
This is useful. It says Newton's method will usually work, if we have a good enough starting guess.

① $F(x) = e^x$



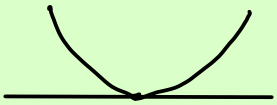
No. No root.

② $F(x) = e^x - 1$



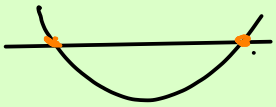
Yes.

③ $F(x) = x^2$



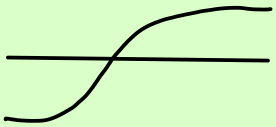
No. F' singular at 0

④ $F(x) = x^2 - 1$



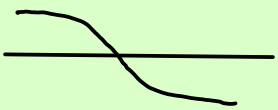
Yes.

⑤ $F(x) = \tanh x$



Yes.

⑥ $F(x) = -\tanh x$



Yes.

⑦ Example ① from last class

$$F\left(\begin{bmatrix} u \\ v \end{bmatrix}\right) = \begin{bmatrix} u^3 - v \\ u^2 + v^2 - 1 \end{bmatrix}$$

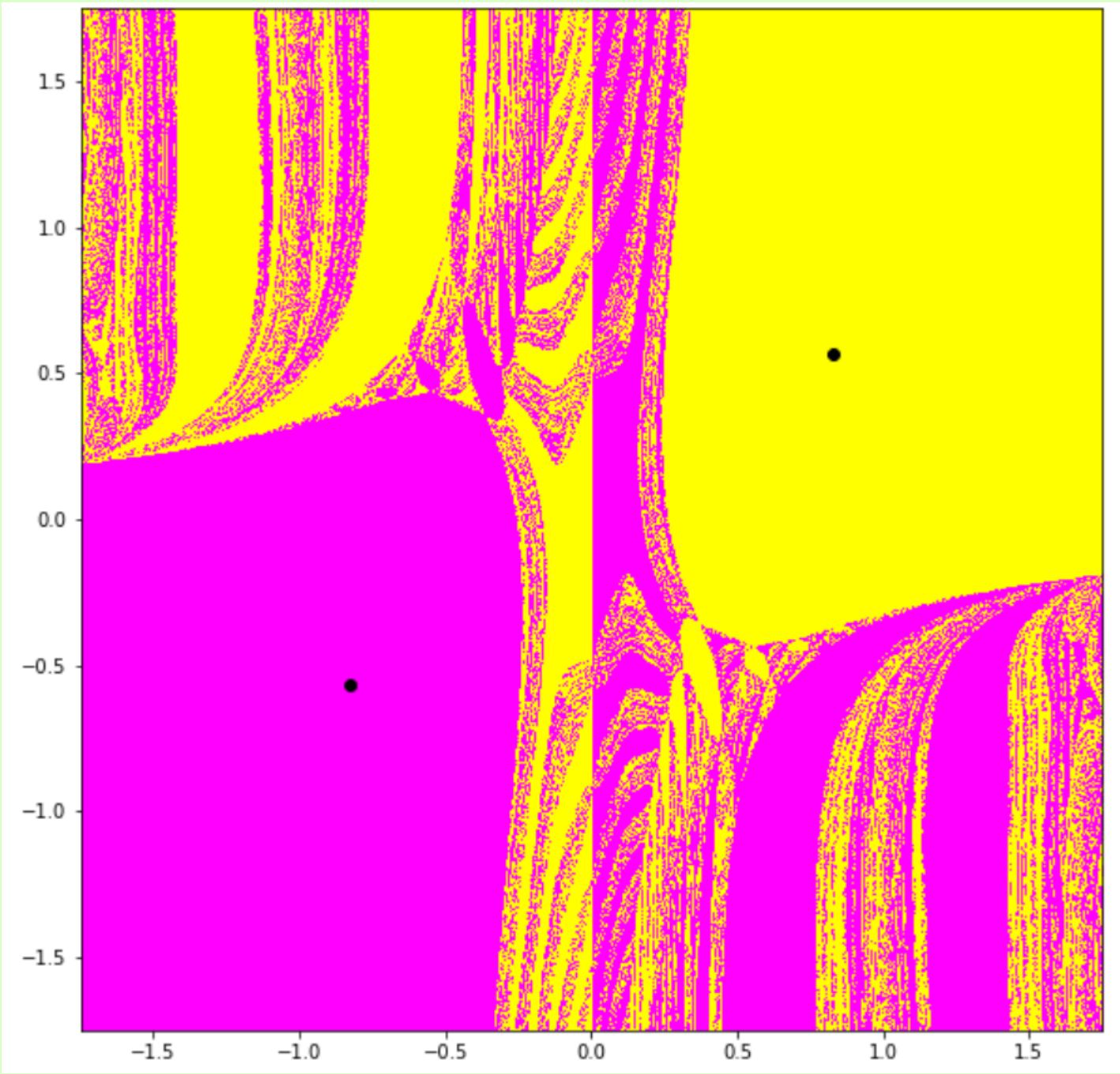
Differentiable everywhere.
Has a root (has 2).

$$F'(u,v) = \begin{matrix} 3u^2 & -1 \\ 2u & 2v \end{matrix}$$

Non-singularity at the root requires computation.
det F' = 6u^2 v + 2u = u(6uv + 2)
Zeros are v-axis, and some hyperbola which is in quad 2&4, so does not contain the roots. If so, theorem applies.

In the last example, each of the two roots has a "basin of attraction" under Newton's method.
What do they look like?

First, I want each of you to guess - make a sketch and upload to UBlearns.
Account for our observation last time that
starting in the 1st quadrant at [.1,.1] Newton converges to the 3rd quadrant root!

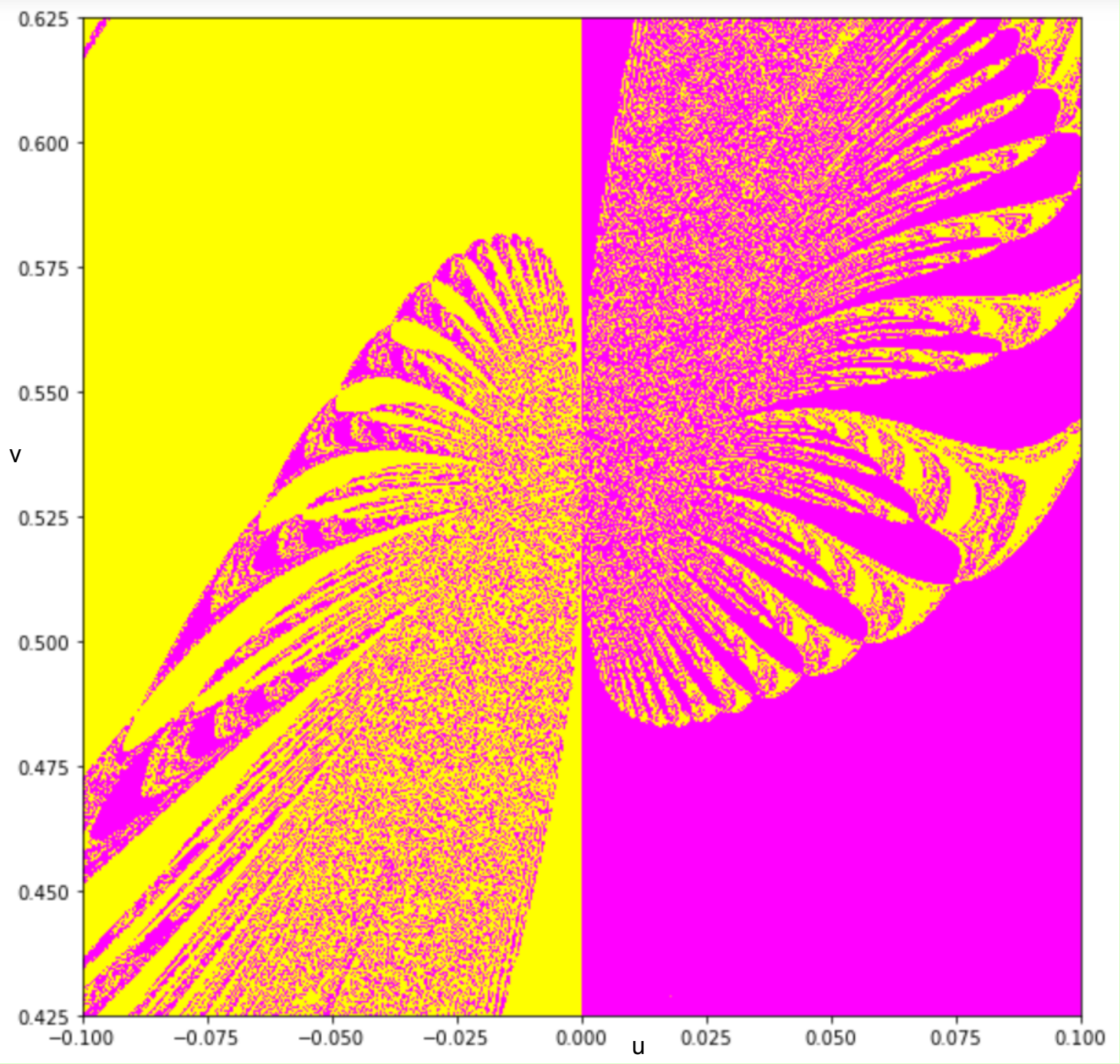


Here is the code that generated the picture above:

```
def newton(F,Fprime,x,tol):
    x = np.array(x,dtype=float)  # floating point copy of x in case it comes in as ints
    nsteps = 0
    while True:
        Fval = F(x)
        Fpval = Fprime(x)
        s = np.linalg.solve(Fpval,-Fval)  # solve for our Newton step
        newx = x + s
        nsteps += 1
        #plt.plot([x[0],newx[0]],[x[1],newx[1]],'k',alpha=0.5)
        #plt.plot(newx[0],newx[1],'ko',alpha=0.2)
        x = newx
        if np.linalg.norm(s) < tol:
            return x,nsteps  # done!

def myF(x):
    u,v = x
    return np.array([ u**3 -v, u**2 + v**2 - 1 ])

def myFprime(x):
    u,v = x
    return np.array([[ 3*u**2, -1 ],
                     [ 2*u,    2*v]])

# first get the two roots
tol = 1e-12
x = [1,.5]
(u1,v1),nits = newton(myF,myFprime,x,tol) # 1st quadrant root
u3,v3 = -u1,-v1                   # 3rd quadrant root
print(u1,v1)

uc,vc = 0,0
r = 1.75
u  = np.linspace(uc-r,uc+r,1000)
v  = np.linspace(vc-r,vc+r,1000)
m = len(u)
basin = np.empty((m,m))
U0,V0 = np.meshgrid(u,v)
#print(U0)
#assert(0)
for i in range(m):
    for j in range(m):
        u = U0[i,j]
        v = V0[i,j]
        try:
            z,nsteps = newton(myF,myFprime,(u,v),1.e-6)
            if True:
                basin[i,j] = z[0]
            else:
                basin[i,j] = np.sqrt(nsteps)
        except:
            basin[i,j] = 0
plt.figure(figsize=(12,10))
plt.imshow(np.flipud(basin),extent=(uc-r,uc+r,vc-r,vc+r),cmap='spring',interpolation='nearest')
plt.plot([u1,u3],[v1,v3],'ko') # plot the two roots
```

Note: this would run a lot faster if "vectorized"
to perform iteration from all the starting points in parallel.
But for clarity of code I did it the slow way.

Let's zoom in on part of this picture:



8.3.2 Convergence Rate of Newton's Method

We now examine the rate of convergence of Newton iteration.

PROPOSITION 8.1

Assume that the hypotheses of Theorem 8.6 hold. Then, for the point of attraction of the Newton iteration (whose existence is guaranteed by Theorem 8.6), we have

$$\lim_{k\to\infty} \frac{\|x^{(k+1)} - x^*\|}{\|x^{(k)} - x^*\|} = 0. \tag{8.30}$$

Moreover, if for some constant $\hat{c}$,

$$\|F'(x) - F'(x^*)\| \le \hat{c}\|x - x^*\| \quad \textit{(that is, } F' \textit{ is Lipschitz continuous)} \tag{8.31}$$

for all x in some neighborhood of x^*, then there exists a positive constant c such that

$$\|x^{(k+1)} - x^{(k)}\| \le c\|x^{(k)} - x^*\|^2. \tag{8.32}$$

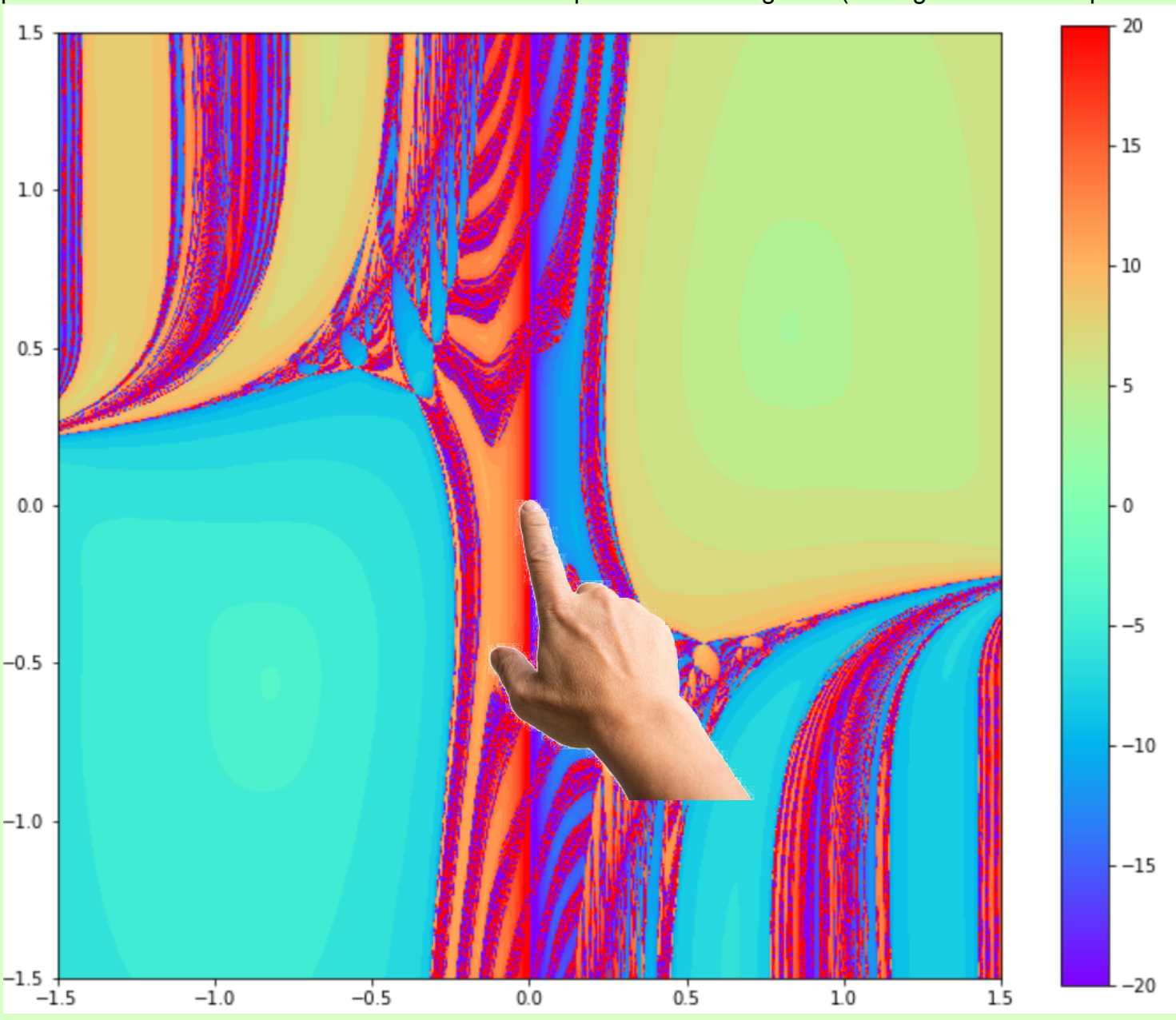
Thus convergence is fast!

REMARK 8.6 If $\|x^{(k+1)} - x^*\|/\|x^{(k)} - x^*\| \le \alpha$ for all k sufficiently large, the convergence is said to be linear. Equation (8.30) indicates Newton's method has superlinear convergence, and if (8.31) is satisfied, then Newton's method is quadratically convergent near x^*. □

And let's see how many iterations it takes to converge from each starting guess in our 2D example:

```
uc,vc = 0,0
r = 1.5
u  = np.linspace(uc-r,uc+r,1000)
v  = np.linspace(vc-r,vc+r,1000)
m = len(u)
basin = np.empty((m,m))
U0,V0 = np.meshgrid(u,v)
nsteps_top = 20

for i in range(m):
    for j in range(m):
        u = U0[i,j]
        v = V0[i,j]
        try:
            z,nsteps = newton(myF,myFprime,(u,v),1.e-6)
            if True:
                basin[i,j] = -min(nsteps,nsteps_top) if z[0]<0 else min(nsteps,nsteps_top)#np.linalg.norm(z-[u1,v
            else:
                basin[i,j] = 0
        except:
            basin[i,j] = 0
plt.figure(figsize=(12,10))
plt.imshow(np.flipud(basin),extent=(uc-r,uc+r,vc-r,vc+r),cmap='rainbow',interpolation='nearest')
plt.colorbar();
```

This picture shows the number of Newton iterations required for convergence (as negative if to 3rd quadrant roo



THEOREM 8.7

(Newton–Kantorovich) Let $F : D \subset \mathbb{R}^n \to \mathbb{R}^n$ be F-differentiable on a convex set $D_0 \subset D$, let $\|\cdot\|$ be some norm, and assume that, for some constant $\gamma \geq 0$,

$$\|F'(x) - F'(y)\| \leq \gamma \|x - y\| \quad \text{for all} \quad x, y \in D_0 \tag{8.38}$$

(That is, assume that F' is Lipschitz continuous on D_0.) Suppose that $F'(x^{(0)})$ is invertible for any $x^{(0)} \in D_0$. Moreover, suppose that, for constants $\beta, \eta > 0$,

$$\|(F'(x^{(0)}))^{-1}\| \leq \beta \tag{8.39}$$

$$\|(F'(x^{(0)}))^{-1} F(x^{(0)})\| \leq \eta. \tag{8.40}$$

Also assume that

$$\alpha = \beta\gamma\eta \leq \frac{1}{2}. \tag{8.41}$$

Set

$$t^* = \frac{1 - (1 - 2\alpha)^{\frac{1}{2}}}{\beta\delta}, \tag{8.42}$$

and assume that

$$\overline{S}(x^{(0)}, t^*) = \left\{ x : \|x - x^{(0)}\| \leq t^* \right\} \subset D_0.$$

Then, the Newton iterates

$$x^{(k+1)} = x^{(k)} - (F'(x^{(k)}))^{-1} F(x^{(k)}), \quad k = 0, 1, 2, \cdots$$

are well-defined, remain in $\overline{S}(x^{(0)}, t^*)$, and converge to a solution x^* of $F(x) = 0$ which is unique in $\overline{S}(x^{(0)}, t^*)$.

Moreover, we have the error estimate

$$\|x^{(k)} - x^*\| \leq \frac{(2\alpha)^{2^k}}{\beta\gamma 2^k} \quad k = 0, 1, 2, \cdots \tag{8.43}$$

This gives us a possible way of calculating how close an initial guess will be close enough for convergence. (t* being the relevant ball radius.)

PROOF See [70].

Quasi-Newton methods that may be more economical

$$x^{(k+1)} = x^{(k)} + s^{(k)}, \quad F'(x^{(k)})s^{(k)} = -F(x^{(k)}). \quad \text{Newton}$$

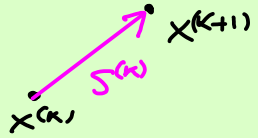
Quasi-Newton methods are modifications of Newton's method for stability or efficiency.

For example, we could use $x^{(k+1)} = x^{(k)} + \lambda s^{(k)}$ with $0 < \lambda < 1$.

which we might call "timid Newton".
In some cases, this enlarges he convergence region.

Or, we could try to avoid the large expense of a full Jacobian evaluation at each step by approximating it based on information we pick up along the way.

Sometimes our main concern is just getting to x*.
Other times our main concern is economy.



If efficiency is important, we could note that evaluating F at x(k) and x(k+1) tells us something about how F varies in the direction from x(k) to x(k+1): it's gives us a secant approximation to the directional derivative along the vector $x^{(k+1)} - x^{(k)} \equiv s^{(k)}$

Now the derivative F' has n^2 components, and this secant information has only n components.
But using the new information

$$\Delta^{(k)} \equiv F(x^{(k+1)}) - F(x^{(k)})$$

allows us to "update" our previous jacobian $F'(x^{(k)})$ to obtain an approximation to $F'(x^{(k+1)})$.

which we'll call $B^{(k+1)}$ (B for Broyden),

In fact, the secant approximation is $\Delta^{(k)} \approx B^{(k+1)} s^{(k)}$.

rise = slope · run

So let's use the secant approximation as a constraint on B(k+1).
That is n scalar constraints on n^2 unknowns.
We need n^2-n additional constraints.

Since the move along s(k) told us NOTHING about the rate of change of F along orthogonal directions, that is along directions

$$w \in S_\perp^{(k)}$$

let's "leave alone" the action of B along $w \in S_\perp^{(k)}$.

That is, require $B^{(k+1)} w = B^{(k)} w \quad \forall w \in S_\perp^{(k)}$.

Since $\dim(S_\perp^{(k)}) = n-1$,

this gives us (n-1)n more scalar constraints on $B^{(k+1)}$: just the number we needed.

The solution of this system of n^2 equations is called the "Broyden rank-1 update" of the jacobian:

$$B^{(k+1)} = B^{(k)} + \frac{\Delta^{(k)} - B^{(k)} s^{(k)}}{s^{(k)T} s^{(k)}} s^{(k)T}$$

We can see by eyeballing it that this formula satisfies the two sets of conditions.

Benefit of Broyden updating

Cheaper than full jacobian evaluation at every step (very much cheaper for large n).

Downside

Because not using the true jacobian, convergence is not quadratic (though it's still superlinear)

Updating inverse of B instead of B: Sherman-Morrison formula

Even with a cheap approximation to F', we still have to solve $B^{(k)} s^{(k)} = -F(x^{(k)})$ for every step $s^{(k)}$.

which costs $\sim \frac{2}{3} n^3$ ops - a lot!

How about updating the inverse of B instead of updating B? Can that be done?

Of course we may need to "get off the ground" by doing a full brute force evaluation of $B^{(0)^{-1}}$ as $F'(x^{(0)})^{-1}$.

So is there a nice cheap formula for $B^{(k+1)^{-1}}$ if $B^{(k+1)} = B^{(k)} + uv^T$ for some u, v?

Yes! The Sherman-Morrison formula is

$$(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1} u v^T A^{-1}}{1 + v^T A^{-1} u}.$$

I will ask you to verify this in the next homework assignment.

Does not require O(n^3) ops.

.

Homotopy (morphing) in root-finding

Often hard to find a good enough starting point $x^{(0)}$ for Newton or quasi-Newton because the "basin of attraction" of the root is very small.

How to come up with a good $x^{(0)}$?

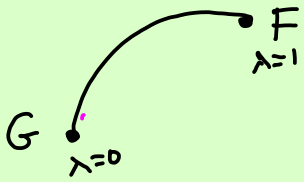
A powerful idea is to consider a family of systems $H: \mathbb{R}^n \times [0,1] \to \mathbb{R}^n$

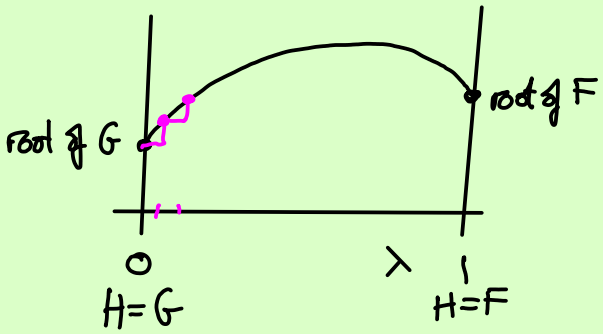
where $H(x,0) = G(x)$, $H(x,1) = F(x)$

$G(x)$: some other "nicer, easier" function

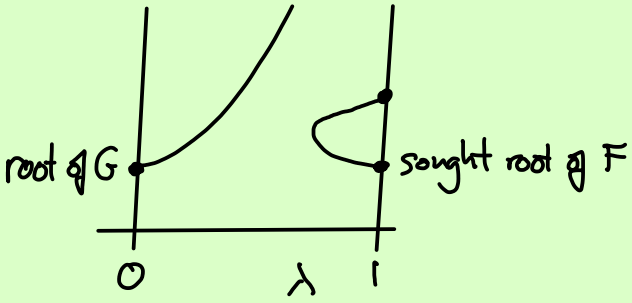
$F(x)$: the function whose root we want

Then what we do is start with an easily obtained root of G, and then gradually "morph" G into F, incrementing the parameter λ in $H(x,\lambda)$, tracking the root as we go, using the root on the previous value of λ as the starting x(0) for the current value of λ, until finally at $\lambda = 1$ we have a root of F.





Choosing a suitable homotopy is not necessarily easy.
A homotopy can easily fail, such as when the root of G and the sought root of F are not connected by a curve, maybe like this:



A lazy choice of homotopy, like G(x) = x - r (really easy to find a root (it's r!)) and $H(x,\lambda) = (1-\lambda)G(x) + \lambda F(x)$ is almost certain to fail.

Knowledge about the particular problem at hand will be useful in constructing a successful homotopy, such as in the toy suspension bridge system of Project Option 3. For example, a good approach might be to let G be a version of F with different, and particularly simplifying, parameter values, and let the homotopy be a linear ramp from the simplifying parameter values to the values you're actually interested in.