Fixed Point for Systems (II) / Newton for systems of equations

Wednesday, September 25, 2024 11:02 AM

Class 13: September 25, 2024

Recall: Last time, we showed that if we have a system of m equations in n real variables x1,...,xn, we can always define a rootfinding problem equivalent to it. We do this by defining a function F: R^n -> R^m where F_i(x1,...,xn) = 0 for i=1,...,m (Fi is obtained by subtracting the right-hand-side from the ith equation). *We will, from now on, assume that m=n, that is, there are as many equations as there are variables.*

We then talked about which methods have a chance of extending to this case, and decided bisection is out, but FPI, Newton and Secant (with some modifications) might still be in. For Fixed Point Iteration, we argued that if I am given the problem F(x) = 0, I can always find G(x) such that G(x) = x is equivalent to F(x) = 0. We came up with the following families of G's:

G(x) = x + cF(x), for c not 0.
G(x) = x + c(x) F(x) for c(x) not 0 around r
G(x) = x + C F(x) for C an invertible n x n matrix
G(x) = x + C(x) F(x) for C(x) invertible n x n matrix (around r)

We then spent some time looking at an implementation and examples of the FPI method for systems, including to solve a problem involving two intersections between circles. We discussed that since x=(x1,...,xn) is a vector and G(x) is a vector, we must use numpy array functions, and vector norms are needed to talk about convergence, error, and so on.

Today, we will wrap up our discussion on how the convergence results generalize to the FPI for systems of equations. Then we will introduce the Newton method for systems.

FPI for systems - we had that:

$\vec{x}_n$ close to r,

1D: $e_{k+1} = g'(r)\, e_k + \frac{1}{2} g''(\xi_k)\cdot e_k^2$

nD: $\underline{\vec{e}_{k+1} = J_G(\vec{v})\cdot \vec{e}_k} + \text{H.O.T } O(\|\vec{e}_k\|^2)$

$\vec{e}_{k+1} \approx J_G(r)\cdot \vec{e}_k \approx J_G^2(r)\, \vec{e}_{k-1} \approx \ldots \approx J_G(r)^k\, \vec{e}_0$

$\boxed{\text{Q: When does } J^k \vec{e}_0 \xrightarrow[k\to\infty]{} \vec{0}\,?\ \left(\|J^k \vec{e}_0\| \to 0\right)}$

Assume that there is a basis of eigenvec $\vec{v}_j$ of J w/ eigenvalues λ_j!

Then $\vec{e}_0 = \alpha_1 \vec{v}_1 + \alpha_2 \vec{v}_2 + \ldots + \alpha_n \vec{v}_n$

$J\vec{e}_0 = \alpha_1 \underbrace{J v_1} + \alpha_2 J v_2 + \ldots + \alpha_n J \vec{v}_n$

$= \alpha_1 \lambda_1 \vec{v}_1 + \alpha_2 \lambda_2 \vec{v}_2 + \ldots + \alpha_n \lambda_n \vec{v}_n$

$$= \alpha_1 \lambda_1 \vec{v}_1 + \alpha_2 \lambda_2 \vec{v}_2 + \dots + \alpha_n \lambda_n \vec{v}_n.$$

$$J^2 \vec{e}_0 = \alpha_1 \lambda_1^2 v_1 + \alpha_2 \lambda_2^2 v_2 + \dots + \alpha_n \lambda_n^2 v_n$$

$$\vdots$$

$$J^k \vec{e}_0 = \alpha_1 \lambda_1^k \vec{v}_1 + \alpha_2 \lambda_2^k \vec{v}_2 + \dots + \alpha_n \lambda_n^k \vec{v}_n$$

Condition is $\boxed{\max\{|\lambda_j|\} < 1}$

→ ask this of $J_G(\vec{x})$ for all $\vec{x}$ in D

→ ask this of $J_G(\vec{r})$ → for x_0 close to r, lin. conv.

"G contractive": G is contractive in $D \subseteq \mathbb{R}^n$ if for all $\vec{x}, \vec{y}$ in D, there is $0 \le L < 1$ s.t.

$$\|G(x) - G(y)\| \le L \, \|x - y\|$$

$$L = \|J_G(r)\|_2 = \max_{\vec{z} \neq 0} \frac{\|J_G(r) z\|_2}{\|z\|_2} < 1$$

NEWTON-RAPHSON

$F(\vec{x}) = 0$, $J_F(\vec{x})$ = $\begin{bmatrix} \frac{\partial F_1}{\partial x_1} & \frac{\partial F_1}{\partial x_2} & \cdots & \frac{\partial F_1}{\partial x_n} \\ \frac{\partial F_2}{\partial x_1} & \frac{\partial F_2}{\partial x_2} & \cdots & \frac{\partial F_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial F_m}{\partial x_1} & \frac{\partial F_m}{\partial x_2} & \cdots & \frac{\partial F_m}{\partial x_n} \end{bmatrix}$

$n \times n$ matrix

$$\begin{bmatrix} F_1(x_1, \dots, x_n) \\ F_2(x_1, \dots, x_n) \\ \vdots \\ F_m(x_1, \dots, x_n) \end{bmatrix}$$

$$\begin{bmatrix} \vdots \\ F_m(x_1,\dots,x_n) \end{bmatrix}$$

$$F(x_1,x_2) = \begin{bmatrix} (x_1-1)^2 + x_2^2 - 1 \\ (x_1-2)^2 + (x_2-1)^2 - 1 \end{bmatrix}$$

$$J_F(x_1,x_2) = \begin{bmatrix} 2(x_1-1) & 2x_2 \\ 2(x_1-2) & 2(x_2-1) \end{bmatrix}$$

1D: $\ell(x) = f(x_0) + f'(x_0)(x-x_0) = 0$

nD: $L(\vec{x}) = F(\vec{x}_0) + J_F(x_0)(\vec{x}-\vec{x}_0) = \vec{0}$

$$\underset{n\times n}{J_F(x_0)}\ \underbrace{(\vec{x}-\vec{x}_0)}_{\text{unknown vector}} = \underbrace{-F(\vec{x}_0)}_{n\times 1}.$$

LINEAR SYSTEM OF EQS: Assume that $J_F(x_0)$ is invertible.

$$(x-x_0) = J_F^{-1}(x_0)(-F(x_0))$$

$$\vec{x}_1 = \vec{x}_0 - J_F^{-1}(x_0)F(x_0)$$

$$\boxed{\vec{x}_{k+1} = \vec{x}_k - J_F^{-1}(\vec{x}_k)F(\vec{x}_k).}$$

$$\boxed{\underline{\text{DO NOT}}\ \text{COMPUTE } J_F^{-1}(x_k)\,!}$$

$\boxed{\text{DO NOT COMPUTE } J_F^{-1}(x_k)\,!}$

MORE EXPENSIVE AND LESS STABLE.

- $\vec{p}_k$ = np.linalg.solve $(J_F(x_k), -F(x_k))$
- $\vec{x}_{k+1} = \vec{x}_k + \vec{p}_k$ . .