CP.2.7.1.b2c, Sauer3

Implement Newton's Method with appropriate starting points to find all solutions. Check with EX.2.7.3 to make sure your answers are correct.

$$\text{(b)} \begin{cases} u^2 + 4v^2 &= 4 \\ 4u^2 + v^2 &= 4 \end{cases} \qquad \text{(c)} \begin{cases} u^2 - 4v^2 &= 4 \\ (u-1)^2 + v^2 &= 4 \end{cases}$$

Hint from Sauer from his solution as in EX.2.7.3:

b. The curves are ellipses with semimajor axes 1 and 2 centered at zero and aligned with the x and y axes. Solving by substitution gives the four solutions

$$\begin{pmatrix} u_1 \\ v_1 \end{pmatrix} = \begin{pmatrix} \frac{2}{\sqrt{5}} \\ \frac{2}{\sqrt{5}} \end{pmatrix}, \quad \begin{pmatrix} u_2 \\ v_2 \end{pmatrix} = \begin{pmatrix} -\frac{2}{\sqrt{5}} \\ \frac{2}{\sqrt{5}} \end{pmatrix}, \quad \begin{pmatrix} u_3 \\ v_3 \end{pmatrix} = \begin{pmatrix} \frac{2}{\sqrt{5}} \\ -\frac{2}{\sqrt{5}} \end{pmatrix}, \quad \begin{pmatrix} u_4 \\ v_4 \end{pmatrix} = \begin{pmatrix} -\frac{2}{\sqrt{5}} \\ -\frac{2}{\sqrt{5}} \end{pmatrix}.$$

c. The curves are a hyperbola and a circle that intersects one half of the hyperbola in two points. Solving by substitution gives the two solutions

$$\begin{pmatrix} u_1 \\ v_1 \end{pmatrix} = \begin{pmatrix} \frac{4}{5}\left(1+\sqrt{6}\right) \\ \frac{1}{5}\sqrt{3+8\sqrt{6}} \end{pmatrix}, \quad \begin{pmatrix} u_2 \\ v_2 \end{pmatrix} = \begin{pmatrix} \frac{4}{5}\left(1+\sqrt{6}\right) \\ -\frac{1}{5}\sqrt{3+8\sqrt{6}} \end{pmatrix}.$$

Special Instructions:

a. Please first compute the two solutions using either Sauer's exact solutions (see Hint above) or `scipy.optimize.fsolve` or `scipy.optimize.root`. This will be useful to compute the forward error.

b. Start Newton's method from a relative distance (as measured by the infinity norm) of at least 0.1 from the solution.

c. At each iteration of Newton's method, you must print:

   (a) k, the iteration number

   (b) the absolute backward error at iteration number k defined by

   $$\|F(x_k)\|_\infty$$

   where x_k is the current iterate.

   (c) the relative forward error at iteration number k defined by

   $$\|x_k - x\|_\infty / \|x\|_\infty$$

   where x_k is the current iterate and x is the solution as computed by `scipy.optimize.fsolve` or `scipy.optimize.root`.

   You can also print the current iterate x_k if you want.