PRACTICE 10: Linear Systems I: Iterative Methods

1 Jacobi Method

1.1 Fixed Point Formulation

Let $A\mathbf{x} = \mathbf{b}$ be a square system of n linear equations. Then:

$$A = D + L + U$$

where:

- D: diagonal part
- L: strictly lower triangular part
- U: strictly upper triangular part

$$D = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{bmatrix}, \quad L = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ a_{21} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & 0 \end{bmatrix}, \quad U = \begin{bmatrix} 0 & a_{12} & \cdots & a_{1n} \\ 0 & 0 & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

Assuming D is invertible, we define:

$$\mathbf{x}_{k+1} = g(\mathbf{x}_k) = B\mathbf{x}_k + \mathbf{c}, \quad \text{with:}$$

Jacobi Iteration Matrix:

$$B = -D^{-1}(L + U),$$

and the free term column being: $\mathbf{c} = D^{-1}\mathbf{b}$.

Equivalence with the Original System:

From $g(\mathbf{x}) = \mathbf{x}$, we derive:

$$-D^{-1}(L + U)\mathbf{x} + D^{-1}\mathbf{b} = \mathbf{x} \Rightarrow (D + L + U)\mathbf{x} = \mathbf{b} \Rightarrow A\mathbf{x} = \mathbf{b}$$

So solving $A\mathbf{x} = \mathbf{b}$ is equivalent to solving $\mathbf{x} = g(\mathbf{x})$.

1.2 Iteration Form

The system of equations

$$\begin{cases} 4x - y + z = 7 \\ 4x - 8y + z = -21 \\ -2x + y + 5z = 15 \end{cases}$$

can be rewritten to isolate each variable:

$$x = \frac{1}{4}(7 + y - z)$$
$$y = \frac{1}{8}(21 + 4x + z)$$
$$z = \frac{1}{5}(15 + 2x - y)$$

This suggests the Jacobi iteration formulas:

$$x^{(k+1)} = \frac{1}{4}(7 + y^{(k)} - z^{(k)})$$
$$y^{(k+1)} = \frac{1}{8}(21 + 4x^{(k)} + z^{(k)})$$
$$z^{(k+1)} = \frac{1}{5}(15 + 2x^{(k)} - y^{(k)})$$

Starting with the initial guess:

$$P_0 = (x_0, y_0, z_0) = (1, 2, 2)$$

the iteration approximates the solution vector step by step.

The following table shows the first few iterations:

Iteration	x_k	y_k	z_k
0	1.0000	2.0000	2.0000
1	1.7500	2.6250	2.0000
2	1.9062	2.9688	2.1000
3	1.9922	3.0352	2.1469
4	2.0455	3.0723	2.1752
5	2.0723	3.0911	2.1895

As the iterations progress, the values get closer to the exact solution $(2, 4, 3)$. The method shows convergence under the strict diagonal dominance condition.

- **Iteration k:** $\mathbf{x}^{(k)} = (x_1^{(k)}, \ldots, x_n^{(k)})$ is the approximation at step k. Then:

The general recurrence relation

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j \neq i} a_{ij} x_j^{(k)} \right), \quad i = 1, \ldots, n$$

The Jacobi iteration can be written explicitly as:

$$\boxed{\mathbf{x}^{(k+1)} = -\underbrace{D^{-1}(L+U)}_{\textbf{Jacobi Matrix}}\mathbf{x}^{(k)} + D^{-1}\mathbf{b}}$$

We assume that the diagonal entries a_{ii} are all nonzero so that the inversion of D makes sense.

1.3 Convergence Condition

Definition: A matrix $A \in \mathbb{R}^{N \times N}$ is *strictly diagonally dominant* if:

$$|a_{kk}| > \sum_{\substack{j=1 \\ j \neq k}}^{N} |a_{kj}| \quad \text{for all } k = 1, 2, \ldots, N$$

This condition ensures convergence of the Jacobi method.

Example: For a 3x3 matrix:

$$\begin{aligned}
&\text{Row 1: } |4| > |-1| + |1| \\
&\text{Row 2: } |-8| > |4| + |1| \\
&\text{Row 3: } |5| > |-2| + |1|
\end{aligned}$$

Theorem (Jacobi Iteration): If A is strictly diagonally dominant, then for any starting vector $\mathbf{x}_0 \in \mathbb{R}^n$, the Jacobi iteration:

$$\mathbf{x}_{k+1} = B\mathbf{x}_k + \mathbf{c}$$

converges to the unique solution $\mathbf{x}$ of $A\mathbf{x} = \mathbf{b}$.

Example 1.1 *Consider the linear system:*

$$\begin{cases} 4x_1 - x_2 + x_3 = 1 \\ 4x_1 - 8x_2 + x_3 = 1 \\ -2x_1 + x_2 + 5x_3 = 1 \end{cases}$$

1. *Verify whether the coefficient matrix A is strictly diagonally dominant.*

2. *If so, solve the system using the Jacobi iterative method.*

3. *Use an initial guess $\boldsymbol{x}^{(0)} = \boldsymbol{0}$, relative tolerance 10^{-6}, and maximum 100 iterations.*

4. *At each step, display the current approximation $\boldsymbol{x}^{(k)}$ and the relative error in the infinity norm:*

$$\frac{\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\|_\infty}{\max(\|\boldsymbol{x}^{(k+1)}\|_\infty, 1)}.$$

MATLAB Code for Jacobi Method

```
% Jacobi method with convergence check
A = [4, -1, 1;    4, -8, 1;   -2, 1, 5];
b = [1; 1; 1];
x0 = zeros(size(b));  % Initial guess
tol = 1e-6;
Nmax = 100;

n = length(b);
D = diag(diag(A));
L = tril(A, -1);
U = triu(A, 1);

% Check for strict diagonal dominance:
for i = 1:n
   if abs(A(i,i)) <= sum(abs(A(i,:))) - abs(A(i,i))
      fprintf("Matrix A is NOT strictly diagonally dominant.\n\n");
      break
   else
      fprintf("Matrix A is strictly diagonally dominant.\n\n");
   end
end

% Jacobi iteration
x = x0;
err=1;
while err>tol && k<Nmax
   xk=x; % initialize xk
   for i = 1:n
      xk(i) = (b(i) - A(i,[1:i-1, i+1:end])*x([1:i-1, i+1:end]))/A(i,i);
   end
   err = norm(xk - x, inf) / max(norm(xk, inf), 1);
   fprintf("Iter %2d: x = [% .6f % .6f % .6f],  rel. error = %.2e\n", ...
   k, xk(1), xk(2), xk(3), err);
   x = xk;
end
%
if err < tol
   fprintf("\nConverged in %d iterations.\n", k);
else
   fprintf("\n Doesn't converge in %d iterations.\n", k);
end
```

For the error we may use

$$\frac{\|\mathbf{x}_{k+1} - \mathbf{x}_k\|}{\|\mathbf{x}_k\|} \leq T$$

as stopping criterion, in the form $\|\mathbf{x}_{k+1} - \mathbf{x}_k\| \leq T \cdot |\mathbf{x}_k\|$.

Example 1.2 *Solve the linear system*

$$\begin{cases} 4x_1 + x_2 - x_3 + 2x_4 = 1 \\ x_1 + 3x_2 + x_3 = 1 \\ x_1 + 2x_2 + 5x_3 + x_4 = 1 \\ x_1 - x_2 + 2x_3 + 6x_4 = 1 \end{cases}$$

using the Jacobi method, with relative tolerance $1\mathrm{e}-6$. *Make sure in advance if the method is applicable and if it is convergent.*

Solution:

we observe that all the elements of the main diagonal of the matrix are nonzero elements but If A was big, it would be better to do:

```
min(abs(diag(A)))
```

Now, we define the Jacobi iteration matrix B, and find it's spectral radius, with $B = -D^{-1}(L+U)$:

```
spectral_radius=max(abs(eig(B)))
```

and, as it is less than 1 (the method is convergent), we can solve it using the Jacobi method.

2 Gauss-Seidel Method

This method, also called *stationary*, updates each component of the solution vector during the iteration as soon as it is computed.

Let us consider a symbolic system of three equations:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 &= b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 &= b_3 \end{aligned}$$

Assuming that a_{11}, a_{22}, a_{33} are nonzero, the iterative scheme is:

$$\begin{aligned} x_1^{(k+1)} &= \frac{1}{a_{11}}\left(b_1 - a_{12}x_2^{(k)} - a_{13}x_3^{(k)}\right) \\ x_2^{(k+1)} &= \frac{1}{a_{22}}\left(b_2 - a_{21}x_1^{(k+1)} - a_{23}x_3^{(k)}\right) \\ x_3^{(k+1)} &= \frac{1}{a_{33}}\left(b_3 - a_{31}x_1^{(k+1)} - a_{32}x_2^{(k+1)}\right) \end{aligned}$$

Note how each newly computed value $x_i^{(k+1)}$ is immediately used in the next equation.

The general recurrence relation

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}\right), \quad i = 1, \ldots, n$$

Splitting the matrix A as in the Jacobi method, the Gauss-Seidel method in matrix form is:

$$\boxed{\mathbf{x}^{(k+1)} = (D - L)^{-1}(U\mathbf{x}^{(k)} + \mathbf{b})}$$

The matrix:

$$G = (D - L)^{-1}U$$

is called the **Gauss - Seidel iteration matrix**.

2.1 Convergence Condition

The Gauss - Seidel method converges to the solution of $A\mathbf{x} = \mathbf{b}$ if the coefficient matrix A is strictly diagonally dominant. That is,

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}|, \quad \text{for all } i.$$

Other sufficient conditions for convergence include:

- A is symmetric and positive definite (SPD).
- The iteration matrix G satisfies $\rho(G) < 1$.

Definition 2.1 *The **spectral radius** of a square matrix $A \in \mathbb{R}^{n\times n}$, denoted by $\rho(A)$, is defined as the maximum of the absolute values of its eigenvalues:*

$$\rho(A) = \max\{|\lambda| : \lambda \textit{ is an eigenvalue of } A\}$$

The spectral radius plays a fundamental role in the convergence analysis of iterative methods. In particular, for an iterative method with iteration matrix G, the method converges if and only if:

$$\rho(G) < 1$$

This ensures that the error $\mathbf{e}^{(k)} = G^k\mathbf{e}^{(0)}$ tends to zero as $k \to \infty$.

Example 2.1 *Solve the following linear system using the Gauss–Seidel method:*

$$\begin{cases} 10x_1 - x_2 + 2x_3 = 6 \\ -x_1 + 11x_2 - x_3 + 3x_4 = 25 \\ 2x_1 - x_2 + 10x_3 - x_4 = -11 \\ 3x_2 - x_3 + 8x_4 = 15 \end{cases}$$

3 Nonlinear systems

Consider a nonlinear system of the form:

$$\mathbf{A}(\mathbf{x}) = \mathbf{b}$$

where

$$\mathbf{A} : \mathbb{R}^n \to \mathbb{R}^m, \qquad \mathbf{x} = (x_1, x_2, \dots, x_n), \qquad \mathbf{A}(\mathbf{x}) = (A_1(\mathbf{x}), A_2(\mathbf{x}), \dots, A_m(\mathbf{x}))$$

This system can be written component-wise as:

$$\begin{cases} A_1(\mathbf{x}) = b_1 \\ A_2(\mathbf{x}) = b_2 \\ \vdots \\ A_m(\mathbf{x}) = b_m \end{cases}$$

Here we distinguish two cases:

- $m = n$: square (determined) system
- $m > n$: overdetermined system

Example

$$\begin{aligned} A_1(x, y) &= x^2 - 2x - y \\ A_2(x, y) &= x^2 + 4y^2 \end{aligned} \qquad \mathbf{b} = (-0.5, 4)$$

So we solve:

$$\begin{cases} A_1(x, y) = 0 \\ A_2(x, y) = 4 \end{cases}$$

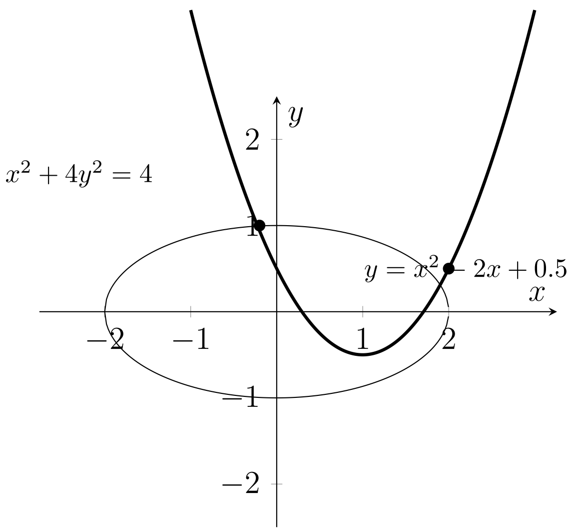


Figure 1: Graphs of the nonlinear system $y = x^2 - 2x + 0.5$ and $x^2 + 4y^2 = 4$.

In this case, the equations implicitly define two curves in the xy-plane. The solution of the nonlinear system is given by their intersection:

- $y = x^2 - 2x + 0.5$: a parabola
- $\dfrac{x^2}{4} + y^2 = 1$: an ellipse

Solution Methods

Two main classes of methods:

- Iterative methods
- Optimization methods

In this example, there are two solutions approximately near the points:

$$(-0.2, 1.0) \quad \text{and} \quad (1.9, 0.3)$$

Newton's Method

In general, the problem is defined as follows:

$$\textbf{Given } \mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m, \textbf{ find } \mathbf{x} \textbf{ such that } \mathbf{f}(\mathbf{x}) = 0.$$

with:

$$\begin{cases} f_1(x_1, x_2, \ldots, x_n) = 0 \\ f_2(x_1, x_2, \ldots, x_n) = 0 \\ \dots\dots\dots\dots\dots\dots\dots \\ f_n(x_1, x_2, \ldots, x_n) = 0 \end{cases}$$

First, denoting by $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, we use the vector function

$$\mathbf{f}(\mathbf{x}) = \begin{pmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ \vdots \\ f_n(\mathbf{x}) \end{pmatrix}$$

The vector function $\mathbf{f}$ is assumed to be *continuous* and *differentiable* in some open set of $\mathbb{R}^n$, with continuous partial derivatives.

The algorithms we study are *iterative.*

◇ **iteration** $k \longrightarrow$ around a point $\mathbf{x}_k$, the function is approximated using a Taylor series expansion truncated after the second-order terms, leading to the following linear system:

$$M_k(\mathbf{x}) = f(\mathbf{x}_k) + J(\mathbf{x}_k)(\mathbf{x} - \mathbf{x}_k),$$

where $J(\mathbf{x}_k)$ is the Jacobian matrix of the system evaluated at $\mathbf{x}_k$:

$$J(\mathbf{x}_k) = \left(\frac{\partial f_i}{\partial x_j}(\mathbf{x}_k)\right) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{pmatrix}$$

Solving the linear system

$$J(\mathbf{x}_k)(\mathbf{x}_{k+1} - \mathbf{x}_k) = -f(\mathbf{x}_k)$$

leads to the recurrence formula:

$$\boxed{\mathbf{x}_{k+1} = \mathbf{x}_k - J(\mathbf{x}_k)^{-1} f(\mathbf{x}_k)}$$

Newton-Raphson Algorithm

1. Define an initial guess $\mathbf{x}_0 \in \mathbb{R}^n$, set $k = 1$ and $\mathbf{x}_k \leftarrow \mathbf{x}_0$.
2. Solve the linear system:

$$J(\mathbf{x}_k)(\mathbf{x}_{k+1} - \mathbf{x}_k) = -f(\mathbf{x}_k)$$

3. If $\|\mathbf{x}_{k+1} - \mathbf{x}_k\|_2 < \text{Tol}$, stop: the problem is solved.
   Otherwise, set $k \leftarrow k + 1$ and go back to step 2.

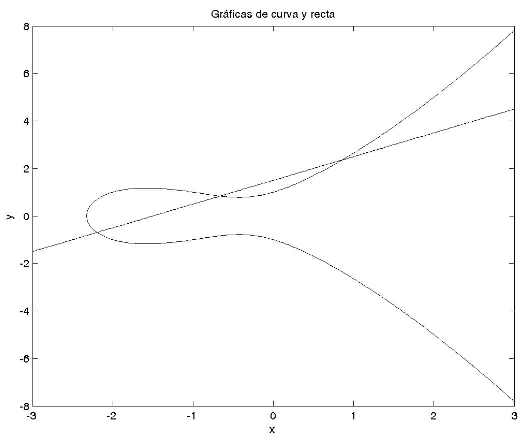
Non Linear Systems - Newton-Raphson Method

```
function [x k]=newton_n(f,J,x0,T,N)
%     function [x k]=newton_n(f,df,x0,T,N)
%     Newton-Raphson method for solving the system f(x)=0
% INPUT ARGUMENTS:
%   f ........ Numerical vector function
%   J ....... Jacobian as numerical function (n x n matrix)
%   x0 ....... Starting guess(n x 1 vector)
%   T ........ Tolerance
%   N ........ Maximum number of iterations
% OUTPUT ARGUMENTS:
%   x ........ Numerical approximation for the solution
%   n ........ Number of iterations done
stopping_criterion=0; k=0; % k is the counter of iterations
while stopping_criterion==0 && k<N
k=k+1;
h=-J(x0)\f(x0);
x=x0+h;
stopping_criterion=norm(h)<=T*norm(x0);
x0=x;
end
if stopping_criterion==0
disp('It does not converge with the requested accuracy.')
disp('The result is not the solution but the last iteration.')
end
```

Example 3.1 *Find the intersection of the curve $y^2 = (x+1)^3 - x$ and the line $y = x + 3/2$, use the tolerance 10^{-6}.*

Solution:

```
ezplot('y^2=(x+1)^3-x',[-3 3 -8 8]); colormap([0 0 0]);
hold on; ezplot('y=x+3/2'); title('Graphs of the curve and the line');
% symbolic vector function
syms x y; f1(x,y)=(x+1)^3-x-y^2; f2(x,y)=x+3/2-y; f(x,y)=[f1(x,y);f2(x,y)]
% jacobian matrix:
df=jacobian(f)
f_num=matlabFunction(f)
f_vector=@(z) f_num(z(1),z(2))
df_num=matlabFunction(df)
df_vector=@(z) df_num(z(1),z(2))
```



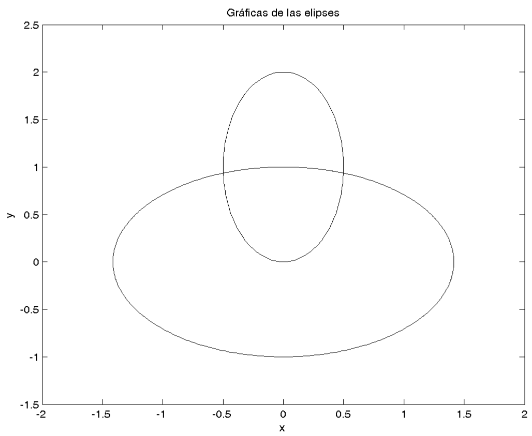
To finish the exercise, we apply the Newton method in [-2;-1], [-1;1] and [1;2]

3.1 The fsolve command

To solve a nonlinear system of n equations and n unknowns, MATLAB has the **fsolve** command. Let's suppose that we have the system defined by the vector function $\mathbf{f}$ then the system is $\mathbf{f}(\mathbf{x}) = \mathbf{0}$, where $\mathbf{x}$ is the vector of unknowns. So, once we define the anonymous vector function $\mathbf{f}$, and the initial guess `v0` (1 x n vector) we can obtain a solution of the nonlinear system by typing `fsolve(f,v0)`.

Example 3.2 *Find the intersection points of the ellipses* $x^2 + 2y^2 = 2$ *and* $4x^2 + (y-1)^2 = 1$.

Solution: First, we plot the graphs.



there are two intersection points. To find them, we do $z_1 = x$, $z_2 = y$, so, if we define

$$\mathbf{z} = (z_1, z_2),\ \ f_1(\mathbf{z}) = z_1^2 + 2z_2^2 - 2,\ \ f_2(\mathbf{z}) = 4z_1^2 + (z_2 - 1)^2 - 1,\ \ \mathbf{f}(\mathbf{z}) = \begin{pmatrix} f_1(\mathbf{z}) \\ f_2(\mathbf{z}) \end{pmatrix}$$

then we already have a vector function which defines the system $\mathbf{f}(\mathbf{z}) = \mathbf{0}$.