

Iterative Methods for Solving Linear Systems

Suzanne Riehl

```
> restart;  
with(LinearAlgebra): with(plots): with(plottools):  
Warning, the name changecoords has been redefined  
Warning, the assigned name arrow now has a global binding
```

Outline

The basic objectives are:

- 1) Learn how to "program" a worksheet.
- 2) Find solutions to systems of equations iteratively using the Jacobi and the Gauss-Siedel methods
- 3) Plot intermediate solutions to see convergence/divergence of method.

```
>
```

A specific example.

```
> eq1:= 7*x-y=5;  
eq2:= 3*x-5*y = -7;
```

$$\begin{aligned}eq1 &:= 7x - y = 5 \\ eq2 &:= 3x - 5y = -7\end{aligned}\tag{1}$$

```
>
```

```
> solve(eq1,x), solve(eq2,y);
```

$$\frac{1}{7}y + \frac{5}{7}, \frac{3}{5}x + \frac{7}{5}\tag{2}$$

Here's a procedure:

```
> nextJacobiPoint:=proc(pair0,eqn1, eqn2)  
local x1,y1:  
x1:=evalf(eval(solve(eq1,x),y=pair0[2])):  
y1 := evalf(eval(solve(eq2,y),x=pair0[1])):  
[x1,y1]:  
end proc;
```

The procedure seems to work if pair0 contains integers. I don't know how to interpret result otherwise.

```
> nextJacobiPoint([5/7,0],eq1,eq2);  
[0.7142857143, 1.828571429]\tag{3}
```

```
>
```

```
> NextPoint := nextJacobiPoint([0,0],eq1,eq2);  
NextPoint:= [0.7142857143, 1.400000000]\tag{4}
```

```
> NextPoint := nextJacobiPoint(NextPoint,eq1,eq2);  
NextPoint:= [0.9142857143, 1.828571429]\tag{5}
```

The following block should generate points, which could then be plotted.

```
>
> listofpoints:=[[27,-13]]:
> for n from 1 to 10
do nextJacobiPoint(listofpoints[n],eq1,eq2):
listofpoints:=[op(listofpoints),%]:
end do:
> listofpoints;
[[27, -13], [-1.142857143, 17.60000000], [3.228571428, 0.7142857142], [
0.8163265306, 3.337142857], [1.191020408, 1.889795918], [0.9842565597,
2.114612245], [1.016373178, 1.990553936], [0.9986505623, 2.009823907], [
1.001403415, 1.999190337], [0.9998843339, 2.000842049], [1.000120293,
1.999930600]]
> PointList := pointplot(listofpoints, connect=true,linestyle=DASH):
LineList := plot({solve(eq1,y),solve(eq2,y)},x=-5..5):
display({PointList, LineList});
```

