4.4.x Random Walk

Random Walk (zufällige Schrittfolge, Zufallsweg) ist ein mathematisches Modell für eine Bewegung, bei der die einzelnen Schritte zufällig erfolgen.

Random-Walk-Modelle eignen sich für nichtdeterministische Zeitreihen oder auch

zur Suche von optimalen Verteilungen bei vielen Parametern.

Beispiel: Aufgabe: RandomWalk im Labor

Methode *walk(obj, steps)*,

die einen Zufallsweg für die Anzahl steps

von Schritten berechnet

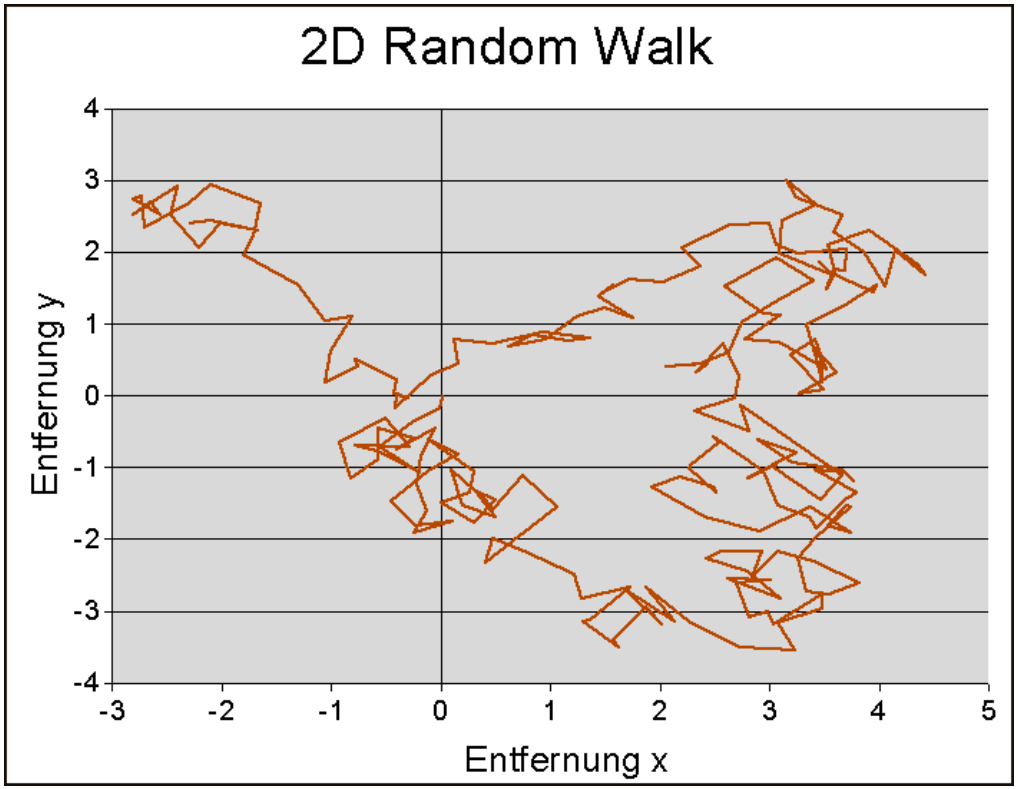
Mit *d=rand(1,2)* erzeugt man pro Schritt

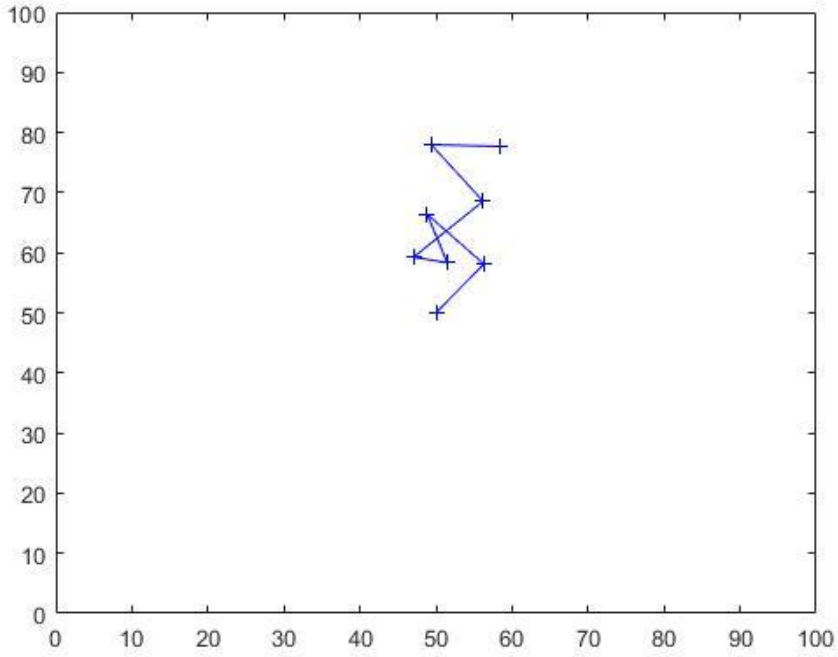
zwei Zufallszahlen im Intervall [0,1].

Nächster Schritt: *Vorgänger + mult * (d - 0.5)*, d - 0.5, damit Intervall [-0.5,+0.5].

Der Parameter *mult* (z.B. mit dem Wert 5)

definiert dabei eine Streckung des Wegs.






```
classdef RandomWalk < handle
  properties
    nZ = 0;          % Zahl der Zeilen de
    nS = 0;          % Zahl der Spalten
    mult = 20;       % Multiplizität der
    start = [0,0];   % Startpunkt
  end
  methods
    % Konstruktor übernimmt Größe des Feldes
    function obj = RandomWalk( nZ, nS )
      . . .
    % steps Schritte laufen, z.B. steps = 1
    function walk( obj, steps )
       . . .
       d = rand( 1, 2 ) - 0.5;    % 2 Zufallszahlen für x und y
       akt = pre + obj.mult*d;    % nächster Punkt
       . . .
```

Aufgabe:

Finde in einem Gebirge den höchsten Punkt!

Dies ist ein 2D-Problem, d.h. eine Variation von zwei Variablen x und y.

In der Praxis hat man oft sehr viel mehr Parameter,

wenn man eine optimale (höchste) Position finden will, z.B. einen Ertrag

im Rahmen der Optimierung eines Systems mit vielen Stellschrauben.

2. Semester:

Finde das Maximum einer Funktion $f(x,y,\ldots)$!

Lösung: Ableiten und Nullsetzen.

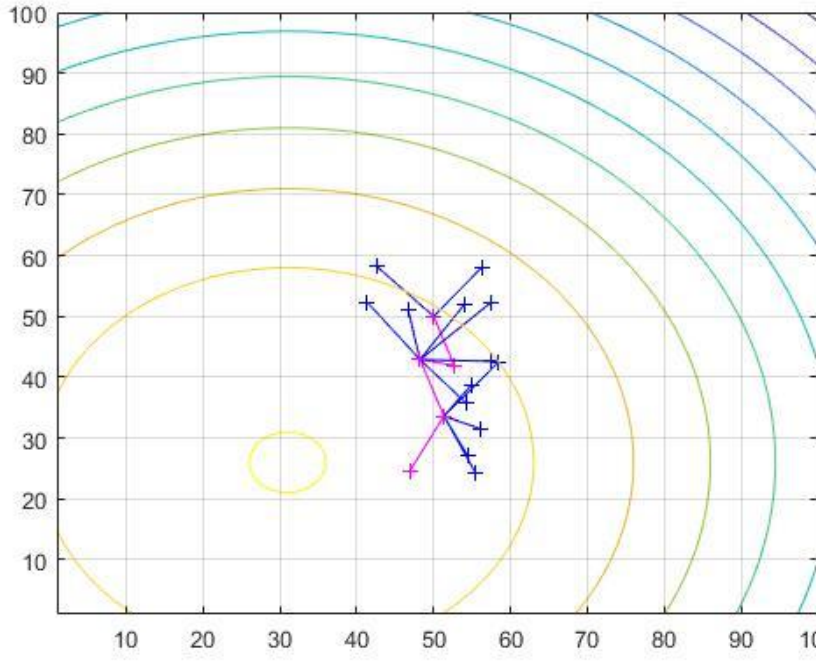
In der Praxis scheitert man damit aber oft,

z.B. weil man die Funktion nach Nullsetzen nicht analytisch lösen kann

oder weil die Funktion nicht differenzierbar ist.

Numerische Lösung: Man probiert verschiedene Werte aus oder
geht in Richtung des stärksten Anstiegs (Gradient).

Problem: Lokale Maxima




```
classdef RandomWalkEval < handle
  properties
    . . .
  end
  methods
    % Konstruktor: Übergabe des Geländes Z
    function obj = RandomWalkEval( Z )
      . . .
    % steps Schritte laufen
    function walk( obj, steps )
      % mit Bewertung, Berg hinauf schreiten:
      val1 = obj.Z(x1,y1); % Höhe des Geländes am vorläufigen Endpunkt
      % neue Höhe liegt tiefer als Starthöhe
      if( val1 < val0 )
        fprintf( 'zurück auf letzte Position \n' );
        . . .
```