

# Line Search Algorithm for Optimization

- Mathe 3 (CES)
- WS24
- Lambert Theisen (theisen@acom.rwth-aachen.de)

```
1 using PlutoUI, Calculus, Gadfly, LinearAlgebra
```

## Define Objective

$$f(x) = x^2$$

```
f = #7 (generic function with 1 method)
```

```
1 f = (x -> x[1]^2)
```

## Line Search

1. Given  $x^{(0)}$
2. For  $k = 0, 1, 2, \dots$  do
  1. Update:  $x^{(k+1)} = x^{(k)} + \alpha_k d^{(k)}$
3. End

```
line_search (generic function with 1 method)
```

```
1 function line_search(f, x0, α, d, kmax)
2     x = x0
3     hist = []
4     push!(hist, x)
5     for k=1:kmax
6         x = x + α(x) * d(x)
7         push!(hist, x)
8     end
9     return x, hist
10 end
```

## Check Line Search

- Observe that different step sizes change the result!

```
☐ (0.0, [1, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0])
```

```
1 line_search(f, 1, (x->1.0), (x->-sign(x)), 10)
```

## Gradient Descent

- Is line search with  $d^{(k)} = -\nabla f(x^{(k)})$

hessian (generic function with 9 methods)

```
1 begin
2   # some notation
3   ∇ = derivative
4   ∇² = hessian
5 end
```

gradient\_descent (generic function with 1 method)

```
1 function gradient_descent(f, x0, α, kmax)
2   return line_search(f, x0, α, (x->-∇(f, x)), kmax)
3 end
```

## Check Gradient Descent

```
☐ (2.03704e-10, [1, 0.8, 0.64, 0.512, 0.4096, 0.32768, 0.262144, 0.209715, 0.167772, ☐ more
```

```
1 gradient_descent(f, 1, (x->0.1), 100)
```

```
☐ (2.65614e-5, [1, -0.9, 0.81, -0.729, 0.6561, -0.59049, 0.531441, -0.478297, 0.430467, ☐ m
```

```
1 gradient_descent(f, 1, (x->0.95), 100) # slower, oscillating but converging
```

## Newton's Method for Optimization

- Is line search with  $d^{(k)} = -[\nabla^2 f(x^{(k)})]^{-1} \nabla f(x^{(k)})$

newton (generic function with 1 method)

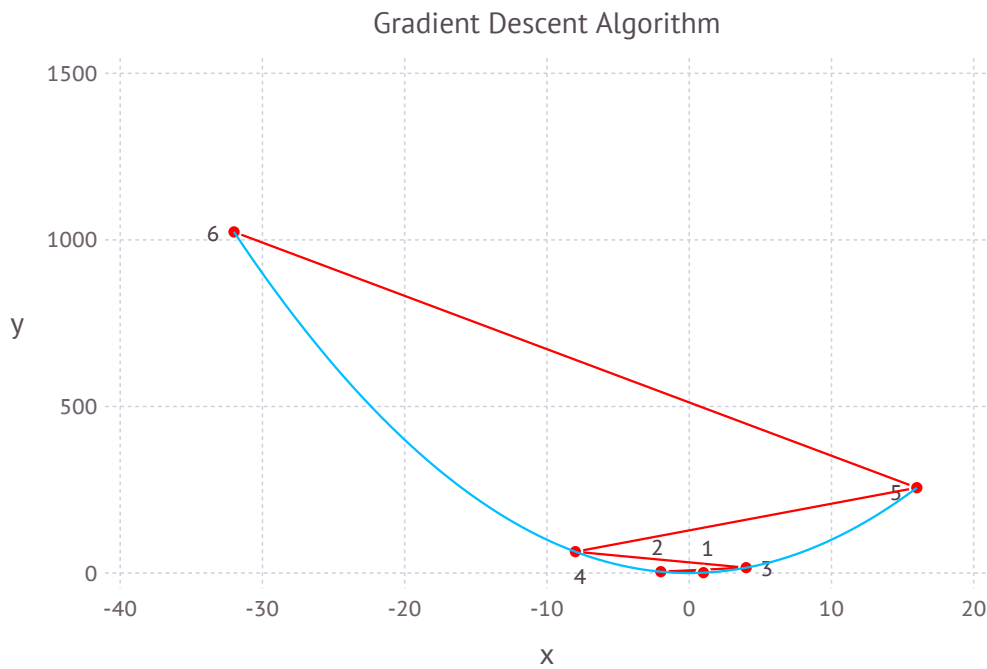
```
1 function newton(f, x0, α, kmax)
2   return line_search(f, x0, α, (x->-inv(∇²(f, x))*∇(f, x)), kmax)
3 end
```

# Check Newton's Method

```
(5.01609e-24, [1.0, -0.500001, 0.25, -0.125, 0.0625001, -0.0312501, 0.015625, -0.00781251
```

```
1 newton(f, 1., (x->1.5), 100) # works well 😎
```

```
(1.26764e30, [1.0, -2.0, 4.00001, -7.99999, 16.0, -31.9999, 63.9998, -128.0, 255.999, 511.999, 1023.999, 2047.999, 4095.999, 8191.999, 16383.999, 32767.999, 65535.999, 131071.999, 262143.999, 524287.999, 1048575.999, 2097151.999, 4194303.999, 8388607.999, 16777215.999, 33554431.999, 67108863.999, 134217727.999, 268435455.999, 536870911.999, 1073741823.999, 2147483647.999, 4294967295.999, 8589934591.999, 17179869183.999, 34359738367.999, 68719476735.999, 137438953471.999, 274877906943.999, 549755813887.999, 1099511627775.999, 2199023255551.999, 4398046511103.999, 8796093022207.999, 17592186044415.999, 35184372088831.999, 70368744177663.999, 140737488355327.999, 281474976710655.999, 562949953421311.999, 1125899906842623.999, 2251799813685247.999, 4503599627370495.999, 9007199254740991.999, 18014398509481983.999, 36028797018963967.999, 72057594037927935.999, 144115188075855871.999, 288230376151711743.999, 576460752303423487.999, 1152921504606846975.999, 2305843009213693951.999, 4611686018427387903.999, 9223372036854775807.999, 18446744073709551615.999, 36893488147419103231.999, 73786976294838206463.999, 147573952589676412927.999, 295147905179352825855.999, 590295810358705651711.999, 1180591620717411303423.999, 2361183241434822606847.999, 4722366482869645213695.999, 9444732965739290427391.999, 18889465931478580854783.999, 37778931862957161709567.999, 75557863725914323419135.999, 151115727451828646838271.999, 302231454903657293676543.999, 604462909807314587353087.999, 1208925819614629174706175.999, 2417851639229258349412351.999, 4835703278458516698824703.999, 9671406556917033397649407.999, 19342813113834066795298815.999, 38685626227668133590597631.999, 77371252455336267181195263.999, 154742504910672534362390527.999, 309485009821345068724781055.999, 618970019642690137449562111.999, 1237940039285380274899124223.999, 2475880078570760549798248447.999, 4951760157141521099596496895.999, 9903520314283042199192993791.999, 19807040628566084398385987583.999, 39614081257132168796771975167.999, 79228162514264337593543950335.999, 158456325028528675187087900671.999, 316912650057057350374175801343.999, 633825300114114700748351602687.999, 1267650600228229401496703205375.999, 2535301200456458802993406410751.999, 5070602400912917605986812821503.999, 10141204801825835211973625643007.999, 20282409603651670423947251286015.999, 40564819207303340847894502572031.999, 81129638414606681695789005144063.999, 162259276829213363391578010288127.999, 324518553658426726783156020576255.999, 649037107316853453566312041152511.999, 1298074214633706907132624082305023.999, 2596148429267413814265248164610047.999, 5192296858534827628530496329220095.999, 10384593717069655257060992658440191.999, 20769187434139310514121985316880383.999, 41538374868278621028243970633760767.999, 83076749736557242056487941267521535.999, 166153499473114484112975882535043071.999, 332306998946228968225951765070086143.999, 664613997892457936451903530140172287.999, 1329227995784915872903807060280344575.999, 2658455991569831745807614120560689151.999, 5316911983139663491615228241121378303.999, 10633823966279326983230456482242756607.999, 21267647932558653966460912964485513215.999, 42535295865117307932921825928971026431.999, 85070591730234615865843651857942052863.999, 170141183460469231731687303715884105727.999, 340282366920938463463374607431768211455.999, 680564733841876926926749214863536422911.999, 1361129467683753853853498429727072845823.999, 2722258935367507707706996859454145691647.999, 5444517870735015415413993718908291383295.999, 10889035741470030830827987437816582766591.999, 21778071482940061661655974875633165533183.999, 43556142965880123323311949751266331066367.999, 87112285931760246646623899502532662132735.999, 174224571863520493293247799005065324265471.999, 348449143727040986586495598010130648530943.999, 696898287454081973172991196020261297061887.999, 1393796574908163946345982392040522594123775.999, 2787593149816327892691964784081045188247551.999, 5575186299632655785383929568162090376495103.999, 11150372599265311570767859136324180752990207.999, 22300745198530623141535718272648361505980415.999, 44601490397061246283071436545296723011960831.999, 89202980794122492566142873090593446023921663.999, 178405961588244985132285746181186892047843327.999, 356811923176489970264571492362373784095686655.999, 713623846352979940529142984724747568191373311.999, 1427247692705959881058285969449495136382746623.999, 2854495385411919762116571938898990272765493247.999, 5708990770823839524233143877797980545530986495.999, 11417981541647679048466287755595961091061972991.999, 22835963083295358096932575511191922182123945983.999, 45671926166590716193865151022383844364247891967.999, 91343852333181432387730302044767688728495783935.999, 182687704666362864775460604089535377456991567871.999, 365375409332725729550921208179070754913983135743.999, 730750818665451459101842416358141509827966271487.999, 1461501637330902918203684832716283019655932542975.999, 2923003274661805836407369665432566039311865085951.999, 5846006549323611672814739330865132078623730171903.999, 11692013098647223345629478661730264157247460343807.999, 23384026197294446691258957323460528314494920687615.999, 46768052394588893382517914646921056628989841375231.999, 93536104789177786765035829293842113257979682750463.999, 187072209578355573530071658587684226515959365500927.999, 374144419156711147060143317175368453031918731001855.999, 748288838313422294120286634350736906063837462003711.999, 1496577676626844588240573268701473812127674924007423.999, 2993155353253689176481146537402947624255349848014847.999, 5986310706507378352962293074805895248510699696029695.999, 11972621413014756705924586149611790497021399392059391.999, 23945242826029513411849172299223580994042798784118783.999, 47890485652059026823698344598447161988085597568237567.999, 95780971304118053647396689196894323976171195136475135.999, 191561942608236107294793378393788647952342390272950271.999, 383123885216472214589586756787577295904684780545900543.999, 766247770432944429179173513575154591809369561091801087.999, 1532495540865888858358347027150309183618739122183602175.999, 3064991081731777716716694054300618367237478244367204351.999, 6129982163463555433433388108601236734474956488734408703.999, 12259964326927110866866776217202473468949912977468817407.999, 24519928653854221733733552434404946937899825954937634815.999, 49039857307708443467467104868809893875799651909875269631.999, 98079714615416886934934209737619787751599303819750539263.999, 196159429230833773869868419475239575503198607639501078527.999, 392318858461667547739736838950479151006397215279002157055.999, 784637716923335095479473677900958302012794430558004314111.999, 1569275433846670190958947355801916604025588861116008628223.999, 3138550867693340381917894711603833208051177722232017256447.999, 6277101735386680763835789423207666416102355444464034512895.999, 12554203470773361527671578846415332832204710888928069025791.999, 25108406941546723055343157692830665664409421777856138051583.999, 50216813883093446110686315385661331328818843555712276103167.999, 100433627766186892221372630771322662657637687111424552206335.999, 200867255532373784442745261542645325315275374222849104412671.999, 401734511064747568885490523085290650630550748445698208825343.999, 803469022129495137770981046170581301261101496891396417650687.999, 1606938044258990275541962092341162602522202993782792835301375.999, 3213876088517980551083924184682325205044405987565585670602751.999, 6427752177035961102167848369364650410088811975131171341205503.999, 12855504354071922204335696738729300820177623950262342682411007.999, 25711008708143844408671393477458601640355247900524685364822015.999, 51422017416287688817342786954917203280710495801049370729644031.999, 102844034832575377634685573909834406561420991602098741459288063.999, 205688069665150755269371147819668813122841983204197482918576127.999, 411376139330301510538742295639337626245683966408394965837152255.999, 822752278660603021077484591278675252491367932816789931674304511.999, 1645504557321206042154969182557350504982735865633579863348609023.999, 3291009114642412084309938365114701009965471731267159726697218047.999, 6582018229284824168619876730229402019930943462534319453394436095.999, 13164036458569648337239753460458804039861886925068638906788872191.999, 26328072917139296674479506920917608079723773850137277813577744383.999, 52656145834278593348959013841835216159447547700274555627155488767.999, 105312291668557186697918027683670432318895095400549111254310977535.999, 210624583337114373395836055367340864637790190801098222508621955071.999, 421249166674228746791672110734681729275580381602196445017243910143.999, 842498333348457493583344221469363458551160763204392890034487820287.999, 1684996666696914987166688442938726917102321526408785780068975640575.999, 3369993333393829974333376885877453834204643052817571560137951281151.999, 6739986666787659948666753771754907668409286105635143120275902562303.999, 13479973333575319897333507543509815336818572211270286240551805124607.999, 26959946667150639794667015087019630673637144422540572481103610249215.999, 53919893334301279589334030174039261347274288845081144962207220498431.999, 107839786668602559178668060348078522694548577690162289924414440996863.999, 215679573337205118357336120696157045389097155380324579848828881993727.999, 431359146674410236714672241392314090778194310760649159697657763987455.999, 862718293348820473429344482784628181556388621521298319395315527974911.999, 1725436586697640946858688965569256363112777243042596638790631055949823.999, 3450873173395281893717377931138512726225554486085193277581262111899647.999, 6901746346790563787434755862277025452451108972170386555162524223799295.999, 13803492693581127574869511724554050904902217944340773110325048447598591.999, 27606985387162255149739023449108101809804435888681546220650096895197183.999, 55213970774324510299478046898216203619608871777363092441300193790394367.999, 110427941548649020598956093796432407239217743554726184882600387580788735.999, 220855883097298041197912187592864814478435487109452369765200775161577471.999, 441711766194596082395824375185729628956870974218904739530401550323154943.999, 883423532389192164791648750371459257913741948437809479060803100646309887.999, 1766847064778384329583297500742918515827483896875618958121606201292619775.999, 3533694129556768659166595001485837031654967793751237916243212402585239551.999, 7067388259113537318333190002971674063309935587502475832486424805170479103.999, 14134776518227074636666380005943348126619871175004951664972849610340958207.999, 28269553036454149273332760011886696253239742350009903329945699220681916415.999, 56539106072908298546665520023773392506479484700019806659891398441363832831.999, 113078212145816597093331040047546785012958969400039613319782796882727665663.999, 226156424291633194186662080095093570025917938800079226639565593765455331327.999, 452312848583266388373324160190187140051835877600158453279131187530910662655.999, 904625697166532776746648320380374280103671755200316906558262375061821325311.999, 1809251394333065553493296640760748560207343510400633813116524750123642650623.999, 3618502788666131106986593281521497120414687020801267626233049500247285301247.999, 7237005577332262213973186563042994240829374041602535252466099000494570602495.999, 14474011154664524427946373126085988481658748083205070504932198000989141204991.999, 28948022309329048855892746252171976963317496166410141009864396001978282409983.999, 57896044618658097711785492504343953926634992332820282019728792003956564819967.999, 115792089237316195423570985008687907853269984665640564039457584007913129639935.999, 231584178474632390847141970017375815706539969331281128078915168015826259279871.999, 463168356949264781694283940034751631413079938662562256157830336031652518559743.999, 926336713898529563388567880069503262826159877325124512315660672063305037119487.999, 1852673427797059126777135760139006525652319754650249024631321344126610074238975.999, 3705346855594118253554271520278013051304639509300498049262642688253220148477951.999, 7410
```



```

1 begin
2   res_gd = gradient_descent(f, 1., (x->1.5), 5)
3   Gadfly.plot(
4     Guide.title("Gradient Descent Algorithm"),
5     layer(f, minimum(res_gd[2]), maximum(res_gd[2])),
6     layer(x=res_gd[2], y=f.(res_gd[2]), label=string.(1:length(res_gd[2])),
7           Geom.point, Geom.path, Geom.label, Theme(default_color=color("red")))
8   )
9 end

```

## Two-Dimensional Optimization

### Define Objective

$$g(x, y) = x^2 + y^2$$

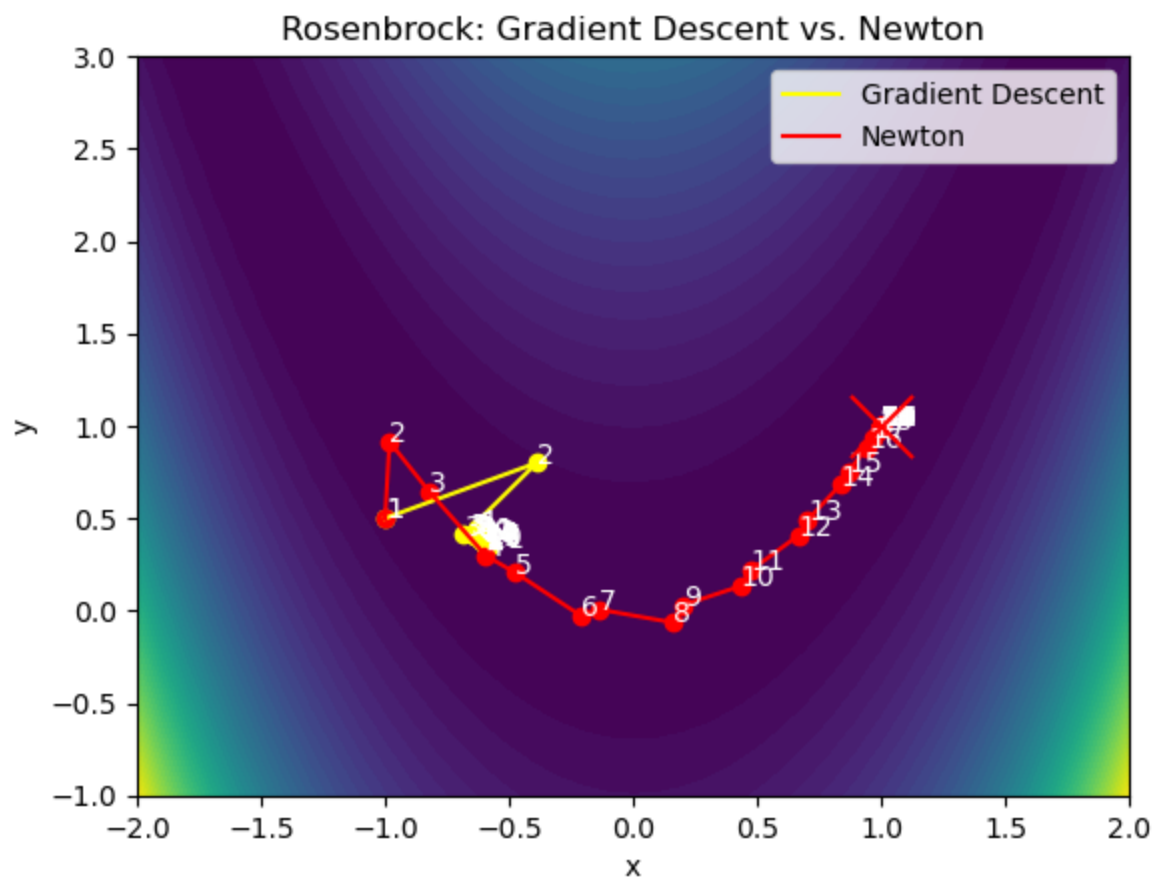
`g = #25 (generic function with 1 method)`

```
1 g = (x->x[1]^2+x[2]^2)
```

### Check Methods

- both work





```
1 begin
2   # Rosenbrock function with  $x^* = [a, a^2]$ ,  $f(x^*)=0$ 
3   a = 1
4   b = 100
5   h = (x -> (a-x[1])^2 + b*(x[2]-x[1]^2)^2)
6
7   x0 = [-1.0, 0.5]
8
9   # Gradient Descent
10  res_gd_2d_rb = gradient_descent(h, x0, (x->0.003), 20)
11  res_gd_2d_rb_x = [res_gd_2d_rb[2][i][1] for i=1:length(res_gd_2d_rb[2])]
12  res_gd_2d_rb_y = [res_gd_2d_rb[2][i][2] for i=1:length(res_gd_2d_rb[2])]
13
14  # Newton
15  res_n_2d_rb = newton(h, x0, (x->0.9), 50)
16  res_n_2d_rb_x = [res_n_2d_rb[2][i][1] for i=1:length(res_n_2d_rb[2])]
17  res_n_2d_rb_y = [res_n_2d_rb[2][i][2] for i=1:length(res_n_2d_rb[2])]
18
19  clf()
20  Δ = 0.1
21  X=collect(-2:Δ:2)
22  Y=collect(-1:Δ:3)
23  F=[h([X[j],Y[i]]) for i=1:length(X), j=1:length(Y)]
24  contourf(X,Y,F, levels=50)
25  PyPlot.title("Rosenbrock: Gradient Descent vs. Newton")
26
27  # res_gd_2d_rb
28  PyPlot.plot(res_gd_2d_rb_x, res_gd_2d_rb_y, color="yellow", label="Gradient
29  Descent")
30  PyPlot.scatter(res_gd_2d_rb_x, res_gd_2d_rb_y, color="yellow")
31  for i=1:length(res_gd_2d_rb_x)
32      annotate(string(i), [res_gd_2d_rb_x[i], res_gd_2d_rb_y[i]], color="w",
33      zorder=2)
34  end
35
36  # res_n_2d_rb
37  PyPlot.plot(res_n_2d_rb_x, res_n_2d_rb_y, color="red", label="Newton")
38  PyPlot.scatter(res_n_2d_rb_x, res_n_2d_rb_y, color="red")
39  for i=1:length(res_n_2d_rb_x)
40      annotate(string(i), [res_n_2d_rb_x[i], res_n_2d_rb_y[i]], color="w",
41      zorder=2)
42  end
43
44  # Legend(["Gradient Descent", "Newton"])
45  legend()
46
47  xlabel("x")
48  ylabel("y")
49
50  # Mark minimum
51  scatter(a, a^2, color="r", s=500, zorder=3, marker="x")
52
53  gcf()
```

51 enu

# Broyden's Method

Homework: Adapt GD and Newton to use the generic framework

line\_search2 (generic function with 1 method)

```

1 function line_search2(f, x0, α, B0, Bk, kmax, tol)
2     x = x0
3     B = B0
4     k = 0
5     Δx = Inf
6     hist = []
7     push!(hist, x)
8     while (k <= kmax) && (norm(Δx) > tol)
9         # invB = length(x)==1 ? 1/B
10        d = -inv(B) * ∇(f, x)
11        Δx = α(x) * d
12        x = x + Δx
13        B = Bk(x, d, f, B)
14        push!(hist, x)
15        k += 1
16    end
17    return x, hist
18 end

```

broyden (generic function with 1 method)

```

1 function broyden(f, x0, α, kmax, tol)
2     return line_search2(
3         f, x0, α,
4         I(length(x0)),
5         (x,d,f,B)->(B + (∇(f, x) * d') / (norm(d,2)^2)), kmax, tol
6     )
7 end

```

☐ `([-1.04412e-22, -1.04412e-22], [[1.0, 1.0], [0.2, 0.2], [-2.64139e-12, -2.64139e-12], [-1.`

1 broyden(g, [1.,1.], (x->0.4), 100, 1E-10) *# works quite fast*

☐ `([2.05496e-22, 2.98015e-22], [[1.0, 1.0], [0.6, 0.6], [0.36, 0.36], [0.216, 0.216], [0.1296`

1 newton(g, [1.,1.], (x->0.4), 100) *# slower than Broyden 🤔*

☐ `([-5.59446e-22], [[1], [0.8], [-6.48059e-11], [-5.59446e-22]])`

1 broyden(f, [1], (x->0.1), 100, 1E-10) *# works*