POPDIP

a POsitive-variables Primal-Dual Interior Point optimization method

Ed Bueler

MATH 661 Optimization

Fall 2024 (version 1.2)

an example of a modern algorithm

- POPDIP is a Newton-type primal-dual interior point algorithm
  - "POPDIP" is just my silly name for it . . . not a thing
- 1990s algorithm mostly covered in standard textbooks
  - section 16.7 in our textbook (Griva, Nash, & Sofer, 2009)
  - chapter 19 in Nocedal & Wright (2006)
- general problem it solves:

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & Ax = b \\ & x \geq 0 \end{array}$$

  - $x \in \mathbb{R}^n$; nonnegativity constraints $x \geq 0$ on *all* variables
  - A is a full row rank $m \times n$ matrix ($m \leq n$), $b \in \mathbb{R}^m$
  - $f(x)$ must be smooth
- user must provide: $f(x)$, $\nabla f(x)$, $\nabla^2 f(x)$, A, b

limitations

- POPDIP can be used as an interior-point method for linear programming (LP)
  - if $f(x) = c^\top x$ then problem is LP standard form
  - but no special performance improvements for LP cases
- it is *not* suitable for:
  - general equality constraints $g_i(x) = 0$
  - general inequality constraints $g_i(x) \geq 0$
  - only a subset of the variables have nonnegativity constraints
  - see section 16.7 of GNS09[1] for how to generalize to such cases

[1] Griva, Nash, and Sofer, *Linear and Nonlinear Optimization*, 2nd ed., SIAM Press 2009

decoding buzzwords

POPDIP is a Newton-type primal-dual interior point algorithm

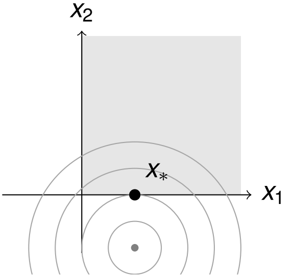
- *Newton-type*: linearize $\nabla f(x)$ and use the Hessian
  - compute p so that $\nabla f(x_k + p) \approx \nabla f(x_k) + \nabla^2 f(x_k) p$ satisfies optimality
- *primal-dual*: keep track of x *and* Lagrange multipliers
  - Lagrangian is $\qquad L(x, \tau, \lambda) = f(x) - \tau^\top (Ax - b) - \lambda^\top x$
  - 1st-order KKT conditions:

$$\begin{aligned} \nabla f(x) - A^\top \tau - \lambda &= 0 \\ -Ax + b &= 0 \\ \lambda_i x_i &= 0, \qquad i = 1, \dots, n \\ x \geq 0, \quad \lambda &\geq 0 \end{aligned}$$

- *interior point*: iterates x_k and λ_k are *both* in the interiors
  - this needs more explanation

example 1: f quadratic, $n = 2$, $m = 0$

- more about the algorithm after 3 example applications
- problem: $\min f(x) = \frac{1}{2}(x_1 - 1)^2 + \frac{1}{2}(x_2 + 1)^2$
  subject to $x \geq 0$
  - $\nabla f(x) = (x_1 - 1, x_2 + 1)^\top$
  - unconstrained minimizer $(1, -1)^\top$
  - solution $x_* = (1, 0)^\top$



- Lagrangian: $L(x, \lambda) = f(x) - \lambda^\top x$
- KKT conditions:

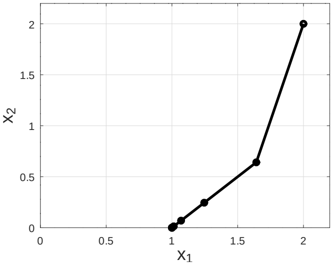
$$
\begin{aligned}
\nabla f(x) - \lambda &= 0 \\
\lambda_i x_i &= 0 \\
x \geq 0, \quad \lambda &\geq 0
\end{aligned}
$$

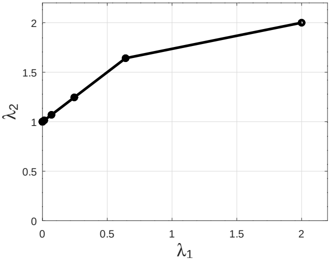
example 1 results (`small.m`)

- result starting from $x_0 = (2, 2)^\top$
- apparent superlinear convergence:

```
>> small
          x_1                 x_2
  0:      2.000000000000000   2.000000000000000
  1:      1.641421356237309   0.641421356237310
  2:      1.245446520462486   0.245446520462486
  3:      1.069404818969903   0.069404818969903
  4:      1.013447008853577   0.013447008853577
  5:      1.000537794151783   0.000537794151783
  6:      1.000000867357066   0.000000867357066
  7:      1.000000000002257   0.000000000002257
  8:      1.000000000000000   0.000000000000000
```

- note complementarity $x_i \lambda_i = 0$ at solution





example 2: f linear, $n = 5$, $m = 3$

- remember this 2D LP problem?

$$\begin{array}{ll} \text{minimize} & z = -x_1 - 2x_2 \\ \text{subject to} & -2x_1 + x_2 \leq 2 \\ & -x_1 + 2x_2 \leq 7 \\ & x_1 \leq 3 \\ & x_1, x_2 \geq 0 \end{array}$$

  - GNS09 section 5.2 example for introducing the simplex method

- convert to standard form by adding slacks:

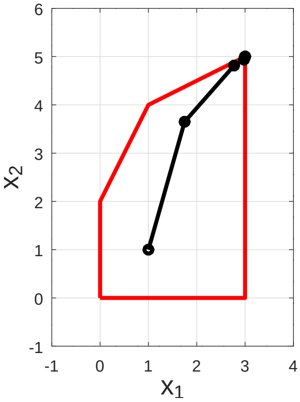
$$\begin{array}{ll} \text{minimize} & z = c^\top x \\ \text{subject to} & Ax = b \\ & x \geq 0 \end{array}$$

  - $A = \begin{bmatrix} -2 & 1 & 1 & 0 & 0 \\ -1 & 2 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix}$, $b = \begin{bmatrix} 2 \\ 7 \\ 3 \end{bmatrix}$, $c = \begin{bmatrix} -1 & -2 & 0 & 0 & 0 \end{bmatrix}^\top$
  - $\nabla f(x) = c, \quad \nabla^2 f(x) = 0$

example 2 results (`linear.m`)

- results starting at $x_0 = (1,1)^\top$:

```
>> linear
         x_1                   x_2
  0:     1.000000000000000     1.000000000000000
  1:     1.749124060578910     3.650175187884218
  2:     2.771966360502739     4.813544496010846
  3:     2.977196636050274     4.937189711618397
  4:     2.993139910878286     4.988863175677374
  5:     2.999412769621014     4.999167116475287
  6:     2.999996717485349     4.999995406776724
  7:     2.999999999901378     4.999999999862158
  8:     3.000000000000000     5.000000000000000
```



- internally, the iteration is happening in 13-dimensional space!
  - $x_k \in \mathbb{R}^5$, $\tau_k \in \mathbb{R}^3$, $\lambda_k \in \mathbb{R}^5$
  - Lagrangian: $L(x,\tau,\lambda) = c^\top x - \tau^\top(Ax-b) - \lambda^\top x$

example 3: f quadratic, n large, $m = 0$

- comes from an *obstacle problem* (continuum):

$$\begin{array}{ll} \text{minimize} & f(u) \\ \text{subject to} & u \geq 0 \end{array}$$

where $f(u) = \int_0^1 \frac{1}{2} u'(x)^2 - q(x)u(x)\,dx$

  - consider only functions with zero end-point values:

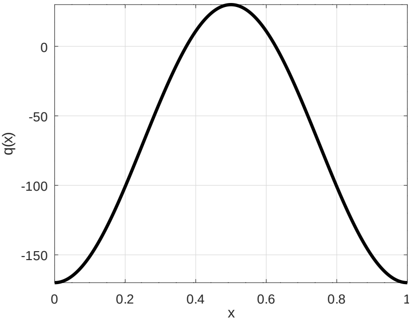
$$S = \{v(x)\ :\ v(0) = v(1) = 0 \text{ and } v(x) \geq 0\}$$

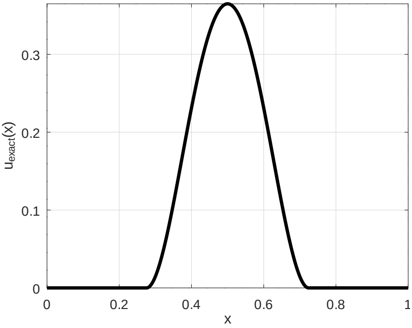
  - ∞-dimensional problem ($n = +\infty$)
  - the "obstacle" is the zero function: $u \geq 0$
  - but no *equality* constraints ($m = 0$)
- discretize using piecewise-linear functions (finite elements):

$$f_n(u) = \Delta x \sum_{i=0}^{n} \frac{1}{2} \left(\frac{u_{i+1} - u_i}{\Delta x} \right)^2 - q(x_{i+\frac{1}{2}}) \frac{u_i + u_{i+1}}{2}$$

example 3: exactly-known continuum solution

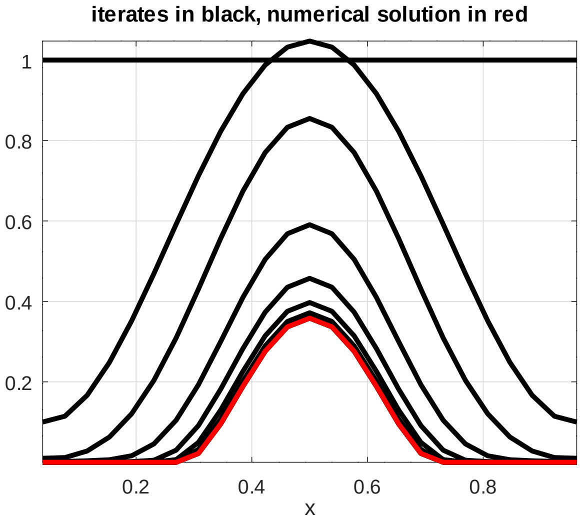
- test using an exactly-known solution to the continuum problem:



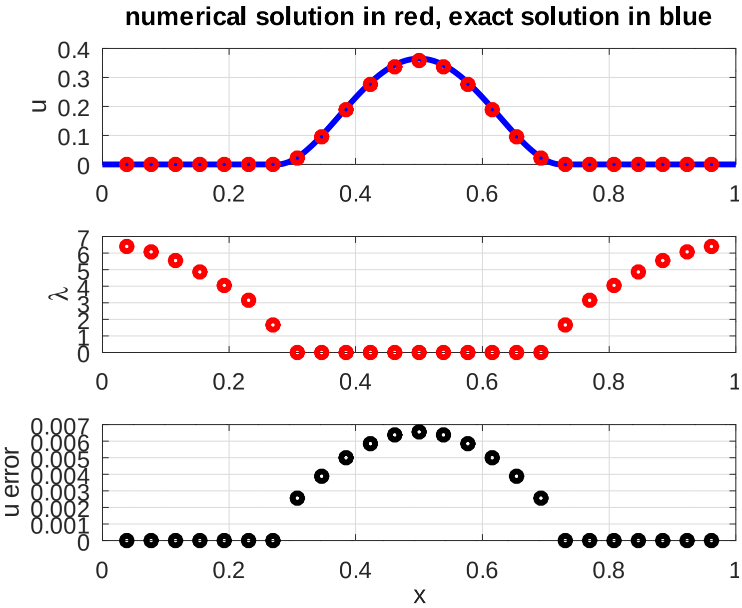


example 3 results (`obstacle.m`)

- results for $n = 25$



example 3 results (`obstacle.m`)



how does POPDIP work?

- easiest to explain it starting *without* equality constraints $Ax = b$
- what are KKT conditions of this problem?

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \geq 0 \end{array}$$

answer: use Lagrangian $L(x, \lambda) = f(x) - \lambda^\top x$; get

$$\begin{array}{rlll} \nabla f(x) - \lambda = 0 & & \textit{stationarity} & (\nabla_x L = 0) \\ x \geq 0 & & \textit{primal feasibility} & \\ \lambda \geq 0 & & \textit{dual feasibility} & \\ x_i \lambda_i = 0 & \text{for } i = 1, \dots, n & \textit{complementarity} & \end{array}$$

- note x and λ are in the same feasible set

$$S = \{v \in \mathbb{R}^n \,:\, v \geq 0\}$$

POPDIP explanation 1: modified KKT conditions

explanation. at each iteration POPDIP approximately solves *modified* KKT conditions which keep the primal/dual iterates x_k, λ_k in the *interior* of S

- let $\mu_k > 0$ be a sequence decreasing to zero
- next values x_{k+1}, λ_{k+1} approximately solve

$$\begin{aligned} \nabla f(x) - \lambda &= 0 && \textit{stationarity} \\ x &\ge 0 && \textit{primal feasibility} \\ \lambda &\ge 0 && \textit{dual feasibility} \\ x_i \lambda_i &= \mu_k && \textit{modified complementarity} \end{aligned}$$

  - $\mu_k > 0$ thus $x_{k+1} > 0$ and $\lambda_{k+1} > 0$
  - inequalities always inactive
  - Newton method: linearize $\nabla f(x)$ at current iterate x_k, λ_k
    - user must supply Hessian $\nabla^2 f(x)$
  - once a search direction is found, the inequality constraints determine how far to move . . .   apply *ratio tests* to x and λ

POPDIP explanation 2: logarithmic barrier

explanation. at each iteration POPDIP approximately solves an *unconstrained* problem on the interior of S, and a *logarithmic barrier* stops the iterates from reaching the boundaries of S

- define a new objective with a logarithmic barrier:

$$\beta_\mu(x) = f(x) - \mu_k \sum_{i=1}^{n} \log x_i$$

  - recall: $\lim_{x \to 0^+} \log x = -\infty$

- solve $\nabla \beta_\mu(x) = 0$: $\quad \nabla f(x) - \mu_k \sum_{i=1}^{n} \frac{1}{x_i} e_i = 0$
- combine symbols: $\lambda_i = \frac{\mu_k}{x_i}$
- get same system as on previous slide:

$$\begin{aligned} \nabla f(x) - \lambda &= 0 \\ x_i \lambda_i &= \mu_k \end{aligned}$$

POPDIP general case: modified KKT

- back to general problem:

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & Ax = b \\ & x \geq 0 \end{array}$$

- Lagrangian: $L(x, \tau, \lambda) = f(x) - \tau^\top (Ax - b) - \lambda^\top x$
- modified KKT:

$$\begin{array}{rl} \nabla f(x) - A^\top \tau - \lambda = 0 & \textit{stationarity} \\ Ax = b & \textit{primal feasibility} \\ x \geq 0 & \textit{primal feasibility} \\ \lambda \geq 0 & \textit{dual feasibility} \\ x_i \lambda_i = \mu_k & \textit{modified complementarity} \end{array}$$

POPDIP general case: linearize and symmetrize

- next iterate:

$$x_{k+1} = x_k + \Delta x, \quad \tau_{k+1} = \tau_k + \Delta\tau, \quad \lambda_{k+1} = \lambda_k + \Delta\lambda$$

- linearize the modified KKT equations:

$$\begin{aligned} \nabla f(x_k) + \nabla^2 f(x_k)\Delta x - A^\top \tau_k - A^\top \Delta\tau - \lambda_k - \Delta\lambda &= 0 \\ -Ax_k - A\Delta x + b &= 0 \\ (\lambda_k)_i (x_k)_i + (x_k)_i (\Delta\lambda)_i + (\lambda_k)_i (\Delta x)_i &= \mu_k \end{aligned}$$

- simplify to smaller, symmetric system:

$$\begin{bmatrix} \nabla^2 f(x_k) + X_k^{-1}\Lambda_k & -A^\top \\ -A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\tau \end{bmatrix} = \begin{bmatrix} -\nabla f(x_k) + A^\top \tau_k + \mu_k X_k^{-1} e \\ Ax_k - b \end{bmatrix}$$

$$\Delta\lambda = \mu_k X_k^{-1} e - \lambda_k - X_k^{-1}\Lambda_k \Delta x$$

  - X_k, Λ_k are diagonal matrices with x_k, λ_k on diagonal
  - $e = (1, \ldots, 1)^\top$

the POPDIP algorithm

1. given $x_0 > 0$
2. determine initial dual variables $\tau_0 = 0$ and $\lambda_0 > 0$
3. for $k = 0, 1, 2, \ldots, \texttt{maxiters} - 1$

$g_k = \nabla f(x_k)$

$\nu_k = \max\{\|g_k - A^\top \tau_k - \lambda_k\|_2, \|b - Ax_k\|_2, \|\Lambda_k x_k\|_2\}$ *merit function*

if $\nu_k < \texttt{atol}$ or $\nu_k < (\texttt{rtol})\,\nu_0$ then stop

$\mu_k = \min\{\theta\nu_k, \nu_k^2\}$ *barrier parameter*

solve for Δx and $\Delta\tau$: *Newton step*

$$\begin{bmatrix} \nabla^2 f(x_k) + X_k^{-1}\Lambda_k & -A^\top \\ -A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\tau \end{bmatrix} = \begin{bmatrix} -g_k + A^\top \tau_k + \mu_k X_k^{-1} e \\ Ax_k - b \end{bmatrix}$$

$\Delta\lambda = \mu_k X_k^{-1} e - \lambda_k - X_k^{-1}\Lambda_k \Delta x$ *part of Newton step*

$\kappa = \max\{\bar{\kappa}, 1 - \nu_k\}$

$\alpha_x = \min_{1\le i\le n}\left\{1, -\kappa\frac{(x_k)_i}{(\Delta x)_i} \,:\, (\Delta x)_i < 0\right\}$ *primal ratio test*

$\alpha_\lambda = \min_{1\le i\le n}\left\{1, -\kappa\frac{(\lambda_k)_i}{(\Delta\lambda)_i} \,:\, (\Delta\lambda)_i < 0\right\}$ *dual ratio test*

$x_{k+1} = x_k + \alpha_x \Delta x$

$\tau_{k+1} = \tau_k + \Delta\tau$

$\lambda_{k+1} = \lambda_k + \alpha_\lambda \Delta\lambda$

trying it yourself

- clone the Github repository:

  `github.com/bueler/popdip`

  - or download a `.zip` release from that website
- download contains documentation PDF `doc.pdf` and these slides `slides.pdf`

trying it yourself

- POPDIP is a MATLAB function:

```
function [x,tau,lam] = popdip(x0,f,A,b)
```

  - input `x0` $\in \mathbb{R}^n$ must have positive entries
  - user-provided function `f` must have signature

    ```
    function [fx,dfx,Hfx] = f(x)
    ```

    where `fx` $= f(x)$, `dfx` $= \nabla f(x)$, `Hfx` $= \nabla^2 f(x)$
  - inputs `A`, `b` are $m \times n$ and $m \times 1$
  - if `A=[]`, `b=[]` then $m = 0$ and equality constraints are ignored
  - output gives last iterate: `x` $\in \mathbb{R}^n$, `tau` $\in \mathbb{R}^m$, `lam` $\in \mathbb{R}^n$
  - see documentation regarding solver parameters
- run examples using driver programs:

```
>> help popdip
>> small                                  % run example 1
>> linear                                 %     example 2
>> obstacle                               %     example 3
```