(Recall that in our original problem the constraint is $2x - y = 5$ or $\begin{pmatrix} 2 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} - 5$, so $A = \begin{pmatrix} 2 & -1 \end{pmatrix}$ and $b = 5$.) By simplifying the above equation, we find that

$$\lambda^{k+1} = (1-\beta)\lambda^k - \beta, \qquad \beta = 5\alpha^k.$$

Back substituting for λ^k in the preceding equation shows that

$$\lambda^{k+1} = (1-\beta)^{k+1}\lambda^0 + (1-\beta)^{k+1} - 1.$$

If $0 < \beta \leq 1$, the preceding line implies that λ^{k+1} converges to $\lambda = -1$, which coincides with the answer provided by the original Lagrangian duality method. Observe that if $\beta = 1$ or $\alpha^k = \frac{1}{5}$, the dual ascent method terminates in one step.

With an appropriate choice of α^k, we have $G(\lambda^{k+1}) > G(\lambda^k)$, so the method makes progress. Under certain assumptions, for example, that J is strictly convex and some conditions of the α^k, it can be shown that dual ascent converges to an optimal solution (both for the primal and the dual). However, the main flaw of dual ascent is that the minimization step may diverge. For example, this happens is J is a nonzero affine function of one of its components. The remedy is to add a penalty term to the Lagrangian.

On the positive side, the dual ascent method leads to a decentralized algorithm if the function J is separable. Suppose that u can be split as $u = \sum_{i=1}^N u_i$, with $u_i \in \mathbb{R}^{n_i}$ and $n = \sum_{i=1}^N n_i$, that

$$J(u) = \sum_{i=1}^N J_i(u_i),$$

and that A is split into N blocks A_i (with A_i a $m \times n_i$ matrix) as $A = [A_1 \ \cdots \ A_N]$, so that $Au = \sum_{k=1}^N A_i u_i$. Then the Lagrangian can be written as

$$L(u, \lambda) = \sum_{i=1}^N L_i(u_i, \lambda),$$

with

$$L_i(u_i, \lambda) = J_i(u_i) + \lambda^\top \left(A_i u_i - \frac{1}{N} b \right).$$

it follows that the minimization of $L(u, \lambda)$ with respect to the primal variable u can be split into N separate minimization problems that can be solved in parallel. The algorithm then performs the N updates

$$u_i^{k+1} = \arg\min_{u_i} L_i(u_i, \lambda^k)$$

in parallel, and then the step

$$\lambda^{k+1} = \lambda^k + \alpha^k (Au^{k+1} - b).$$