Nonlinear Programming Methods.S1
Separable Programming

Separable programming is important because it allows a convex nonlinear program to be approximated with arbitrary accuracy with a linear programming model. The idea is to replace each nonlinear function with a piecewise linear approximation. Global solutions can then be obtained with any number efficient LP codes. For nonconvex problems, the approach is still valid but more work needs to be done. Either a mixed-integer linear programming problem must be solved as discussed in Section 8.8, or a modified version of the simplex algorithm with a limited basis entry rule can be applied to the model directly. The candidates for the entering variable must be restricted to maintain the validity of the LP approximation. In this case a local optimum is obtained but it is possible to find the global optimum with the help of branch and bound.

Problem Statement

Consider again the general nonlinear programming problem

$$\text{Minimize}\{f(\mathbf{x}) : g_i(\mathbf{x}) \le b_i\ ,\ i = 1,\ldots,m\}$$

with two additional provisions: (*i*) the objective function and all constraints are *separable*, and (*ii*) each decision variable x_j is bounded below by 0 and above by a known constant u_j, $j = 1,\ldots,n$. Recall that a function, $f(\mathbf{x})$, is separable if it can be expressed as the sum of functions of the individual decision variables.

$$f(\mathbf{x}) = \sum_{j=1}^{n} f_j(x_j)$$

The separable nonlinear programming problem has the following structure.

$$\begin{aligned} \text{Minimize} \quad & \sum_{j=1}^{n} f_j(x_j) \\ \text{subject to} \quad & \sum_{j=1}^{n} g_{ij}(x_j) \le b_i,\ i = 1,\ldots,m \\ & 0 \le x_j \le u_j,\ \ j = 1,\ldots,n \end{aligned}$$

The key advantage of this formulation is that the nonlinearities are mathematically independent. This property in conjunction with the finite bounds on the decision variables permits the development of a piecewise linear approximation for each function in the problem.

Linearization

Consider the general nonlinear function $f(x)$ depicted in Fig. 5. To form a piecewise linear approximation using, say, r line segments, we must first select $r+1$ values of the scalar x within its range $0 \le x \le u$ (call them $\bar{x}_0, \bar{x}_1, \ldots, \bar{x}_r$) and let $f_k = f(\bar{x}_k)$ for $k = 0,1,\ldots,r$. At the boundaries we have $\bar{x}_0 = 0$ and $\bar{x}_r = u$. Notice that the values of $\bar{x}_k$ do not have to be evenly spaced.

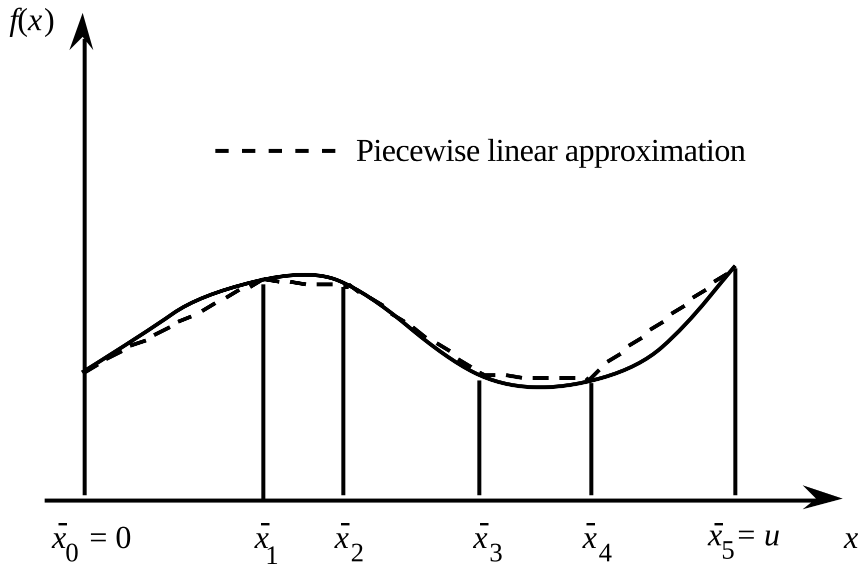

Figure 5. Piecewise linear approximation of a nonlinear function

Recall that any value of x lying between the two endpoints of the kth line segment may be expressed as

$$x = \alpha\bar{x}_{k+1} + (1-\alpha)\bar{x}_k \quad \text{or} \quad x - \bar{x}_k = \alpha(\bar{x}_{k+1} - \bar{x}_k) \text{ for } 0 \le \alpha \le 1.$$

where $\bar{x}_k$ ($k = 0,1,\ldots,r$) are data and α is the decision variable. This relationship leads directly to an expression for the kth line segment.

$$\hat{f}(x) = f_k + \frac{f_{k+1} - f_k}{\bar{x}_{k+1} - \bar{x}_k}(x - \bar{x}_k) = \alpha f_{k+1} - (1-\alpha)f_k \text{ for } \quad 0 \le \alpha \le 1$$

The approximation $\hat{f}(x)$ becomes increasingly more accurate as r gets larger. Unfortunately, there is a corresponding growth in the size of the resultant problem.

For the kth segment, let $\alpha = \alpha_{k+1}$ and $(1 - \alpha) = \alpha_k$. As such, for $\bar{x}_k \leq x \leq \bar{x}_{k+1}$, the above expresses is

$$x = \alpha_{k+1}\bar{x}_{k+1} + \alpha_k \bar{x}_k \text{ and } \hat{f}(x) = \alpha_{k+1} f_{k+1} + \alpha_k f_k$$

where $\alpha_k + \alpha_{k+1} = 1$ and $\alpha_k \geq 0$, $\alpha_{k+1} \geq 0$. Generalizing this procedure to cover the entire range over which x is defined gives

$$x = \sum_{k=0}^{r} \alpha_k \bar{x}_k, \quad \hat{f}(x) = \sum_{k=0}^{r} \alpha_k f_k, \quad \sum_{k=0}^{r} \alpha_k = 1, \ \alpha_k \geq 0, \ k = 0,\ldots,r$$

such that at least one and no more than two α_k can be greater than zero. Furthermore, we require that if two α_k are greater than zero, their indices must differ by exactly 1. In other words, if α_s is greater than zero then only one of either α_{s+1} or α_{s-1} can be greater than zero. If the last condition, known as the *adjacency criterion*, is not satisfied, the approximation to $f(x)$ will not lie on $\hat{f}(x)$.

To apply the above transformations, a grid of $r_j + 1$ points must be defined for each variable x_j over its range. This requires the use of an additional index for each variable and function. For the jth variable, for example, $f_j + 1$ data points result: $\bar{x}_{1j}, \bar{x}_2, \ldots, \bar{x}_{r_j j}$. With this notation in mind, the separable programming problem in **x** becomes the following "almost" linear program in α.

$$\text{Minimize } f(\alpha) = \sum_{j=1}^{n} \sum_{k=0}^{r_j} \alpha_{kj} f_{kj}(\bar{x}_{kj})$$

$$\text{subject to } g_i(\alpha) = \sum_{j=1}^{n} \sum_{k=0}^{r_j} \alpha_{kj} g_{kij}(\bar{x}_{kj}) \leq b_i \, , \ i = 1,\ldots,m$$

$$\sum_{k=0}^{r_j} \alpha_{kj} = 1, \ j = 1,\ldots,n$$

$$\alpha_{kj} \geq 0,\ k = 0,\ldots,r_j,\ \ j = 1,\ldots,n.$$

The reason that this is an "almost" linear programming problem is that the adjacency criterion must be imposed on the new decision variables α_{kj} when any of the functions are nonconvex. This can be accomplished with a restricted basis entry rule. When all the functions are convex, the adjacency criterion will automatically be satisfied so no modifications of the simplex algorithm are necessary. Note that the approximate problem has $m + n$ constraints and $\Sigma_j r_j + n$ variables.

From a practical point of view, one might start off with a rather large grid and find the optimum to the corresponding approximate problem. This should be easy to do but the results may not be very accurate. To improve on the solution, we could then introduce a smaller grid in the neighborhood of the optimum and solve the new problem. This idea is further discussed by Bard et al. (2000).

Example 13

Consider the problem below whose feasible region is shown graphically in Fig. 6. All the functions are convex but the second constraint is $g_2(\mathbf{x}) \geq 10$. This implies that the feasible region is not convex so the solution to the approximate problem may not be a global optimum.

$$\begin{aligned}
\text{Minimize} \quad & f(\mathbf{x}) = 2x_1^2 - 3x_1 + 2x_2 \\
\text{subject to} \quad & g_1(\mathbf{x}) = 3x_1^2 + 4x_2^2 \leq 8 \\
& g_2(\mathbf{x}) = 3(x_1 - 2)^2 + 5(x_2 - 2)^2 \geq 10 \\
& g_3(\mathbf{x}) = 3(x_1 - 2)^2 + 5(x_2 - 2)^2 \leq 21 \\
& 0 \leq x_1 \leq 1.75,\ 0 \leq x_2 \leq 1.5
\end{aligned}$$

The upper bounds on the variables have been selected to be redundant. The objective function and constraints are separable with the individual terms being identified in Table 3.

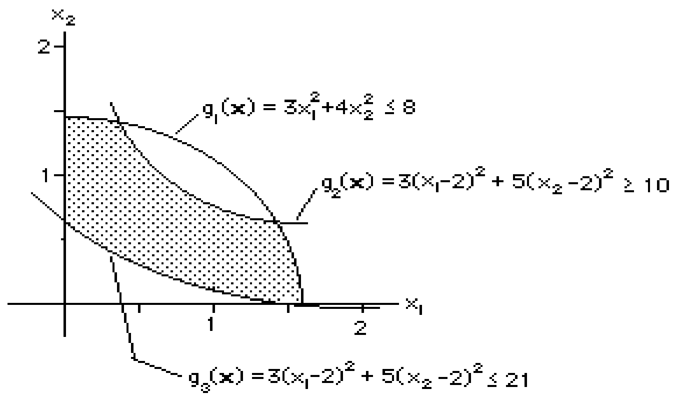

Figure 6. Feasible region for example

Table 3. Separable functions for example

j	1	2
$f_j(x_j)$	$2x_1^2 - 3x_1$	$2x_2$
$g_{1j}(x_j)$	$3x_1^2$	$4x_2^2$
$g_{2j}(x_j)$	$3(x_1 - 2)^2$	$5(x_2 - 2)^2$
$g_{3j}(x_j)$	$3(x_1 - 2)^2$	$5(x_2 - 2)^2$

To develop the piecewise linear approximations we select six grid points for each variable and evaluate the functions at each point. The results are given in Table 4. For this example $n = 2$, $m = 3$, $r_1 = 5$, and $r_2 = 5$. As an illustration, the piecewise linear approximations of $f(x_1)$ and $g_{12}(x_2)$ along with the original graphs are depicted in Fig. 7. The full model has 5 constraints and 12 variables. The coefficient matrix is given in Table 5 where the last two rows correspond to the summation constraints on the two sets of α variables.

Table 4. Grid points and corresponding function values

k	0	1	2	3	4	5
x_{k1}	0	0.4	0.75	1.0	1.25	1.75
x_{k2}	0	0.3	0.6	0.9	1.2	1.5
f_{k1}	0	– 0.88	– 1.125	– 1	– 0.625	0.875
g_{k11}	0	0.48	1.6875	3	4.6875	9.1875
g_{k21}	12	7.68	4.6875	3	1.6875	0.1875
g_{k31}	12	7.68	4.6875	3	1.6875	0.1875
f_{k2}	0	0.6	1.2	1.8	2.4	3
g_{k12}	0	0.36	1.44	3.24	5.76	9
g_{k22}	20	14.45	9.8	6.05	3.2	1.25
g_{k32}	20	14.45	9.8	6.05	3.2	1.25

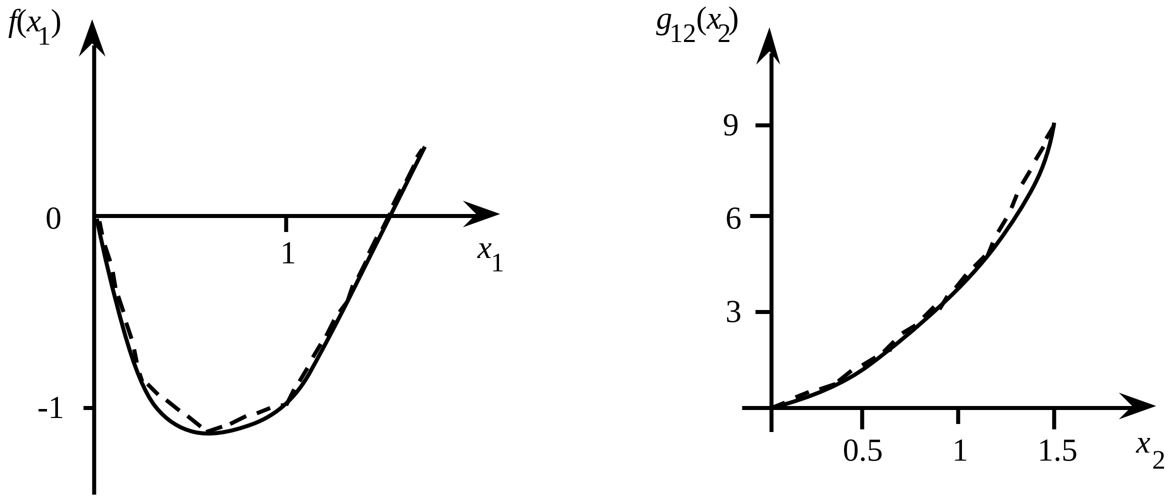

Figure 7. Piecewise linear approximations for $f(x_1)$ and $g_{12}(x_2)$

Table 5. Coefficients of the linear programming model

	α_{01}	α_{11}	α_{21}	α_{31}	α_{41}	α_{51}	α_{02}	α_{12}	α_{22}	α_{32}	α_{42}	α_{52}
f	0	0.88	1.125	1	0.625	- 0.875	0	- 0.6	- 1.2	- 1.8	- 2.4	-3
g_1	0	0.48	1.6875	3	4.6875	9.1875	0	0.36	1.44	3.24	5.76	9
g_2	12	7.68	4.6875	3	1.6875	0.1875	20	14.45	9.8	6.05	3.2	1.25
g_3	12	7.68	4.6875	3	1.6875	0.1875	20	14.45	9.8	6.05	3.2	1.25
x_1	1	1	1	1	1	1	0	0	0	0	0	0
x_2	0	0	0	0	0	0	1	1	1	1	1	1

The problem will be solved with a linear programming code modified to enforce the adjacency criterion. In particular, for the *j*th variable we do not allow an α_{kj} variable to enter the basis unless $\alpha_{k-1,j}$ or $\alpha_{k+1,j}$ is already in the basis, or no α_{kj} ($k = 0,1, \ldots, 5$) is currently basic. The following slack and artificial variables will be used to put the problem into standard simplex form.

s_1 = slack for constraint 1, g_1

s_2 = surplus for constraint 2, g_2

a_2 = artificial for constraint 2, g_2

s_3 = slack for constraint 3, g_3

a_4 = artificial for constraint 4, $\Sigma_k \alpha_{k1}$

a_5 = artificial for constraint 5, $\Sigma_k \alpha_{k2}$

The initial basic solution is

$$\mathbf{x}_B = (s_1, a_2, s_3, a_4, a_5) = (8, 10, 21, 1, 1)$$

which is seen to contain three artificial variables. The phase 1 procedure required five iterations to drive the objective function to zero. Phase 2 also required five iterations. The corresponding data are presented Table 6. The most negative rule was used to select the entering variable but if that variable led to a violation of the adjacency criterion the next one on the list was selected. For example, at iteration 3.1, α_{21} is selected to enter the basis but since it is not adjacent to α_{51} it is not permitted to do so. The nonbasic variable with the next smallest reduced cost is α_{11} but it too does not satisfy the adjacency criterion. Finally, at iteration 3.4 the variable with the smallest reduced cost not yet examined is α_{41} which meets the criterion and so is allowed to enter the basis. A pivot is executed with a_4 the leaving variable.

Table 6. Iterations of restricted entry simplex algorithm

Iter.	Entering	Leaving	Basic variables
0			s_1, a_2, s_3, a_4, a_5
1	α_{02}	a_2	$s_1, \alpha_{02}, s_3, a_4, a_5$
2	α_{51}	s_1	$\alpha_{51}, \alpha_{02}, s_3, a_4, a_5$
3.1	α_{21}	—	not adjacent to α_{51}
3.2	α_{11}	—	not adjacent to α_{51}
3.3	α_{31}	—	not adjacent to α_{51}
3.4	α_{41}	a_4	$\alpha_{51}, \alpha_{02}, s_3, \alpha_{41}, a_5$
4.1	α_{52}	—	not adjacent to α_{02}
4.2	α_{42}	—	not adjacent to α_{02}
4.3	α_{32}	—	not adjacent to α_{02}
4.4	α_{22}	α_{02}	$\alpha_{51}, \alpha_{22}, s_3, \alpha_{41}, a_5$
5.1	α_{52}	—	not adjacent to α_{22}
5.2	α_{42}	—	not adjacent to α_{22}
5.3	α_{32}	a_5	$\alpha_{51}, \alpha_{22}, s_3, \alpha_{41}, \alpha_{32}$
Begin phase 2			
5.3			$\alpha_{41}, \alpha_{32}, s_3, \alpha_{51}, \alpha_{22}$
5.4	α_{02}	—	not adjacent to α_{22}
5.5	α_{42}	—	not adjacent to α_{22}
5.6	s_1	α_{51}	$\alpha_{41}, \alpha_{32}, s_3, s_1, \alpha_{22}$
6	α_{31}	α_{41}	$\alpha_{31}, \alpha_{32}, s_3, s_1, \alpha_{22}$
7	s_2	α_{32}	$\alpha_{31}, s_2, s_3, s_1, \alpha_{22}$
8.1	α_{02}	—	not adjacent to α_{22}
8.2	α_{12}	α_{22}	$\alpha_{31}, s_2, s_3, s_1, \alpha_{12}$
9	α_{02}	s_3	$\alpha_{31}, s_2, \alpha_{02}, s_1, \alpha_{12}$

At the end of iteration 9, the algorithm terminates with the optimal solution

$s_1 = 4.87$, $s_2 = 11$, $\alpha_{31} = 1$, $\alpha_{02} = 0.6396$, $\alpha_{12} = 0.3604$ and $\hat{f} = -0.7838$.

In terms of the original problem variables, this corresponds to

$$x_1 = \alpha_{31}\bar{x}_{31} = 1 \text{ and } x_2 = \alpha_{02}\bar{x}_{02} + \alpha_{12}\bar{x}_{12} = 0.1081.$$

When comparing these values with the true optimum $x_1^* = 0.9227$, $x_2^* = 0.1282$, and $f(\mathbf{x}^*) = -0.8089$, we can observe the effect of the linearization process on the accuracy of the solution. With respect to the objective function, our six-point approximation is off by 3.2%. For this particular problem, however, we can get as close as desired by making the grid size arbitrarily small in the neighborhood of the true optimum.

It is interesting to note that while the basic solutions given in Table 6 lie at vertices in the α space, they do not necessarily have any relation to the vertices in the original **x** space. This is illustrated in Fig. 8 which plots the feasible region and several isovalue contours of $f(\mathbf{x})$. Also shown are the 6 basic solutions that comprise phase 2. The first basic solution corresponds to iteration 5.3 in Table 6. As can be seen, none of the 6 iterates in the figure lies at a vertex of the feasible region. At the optimum only g_3 is binding.

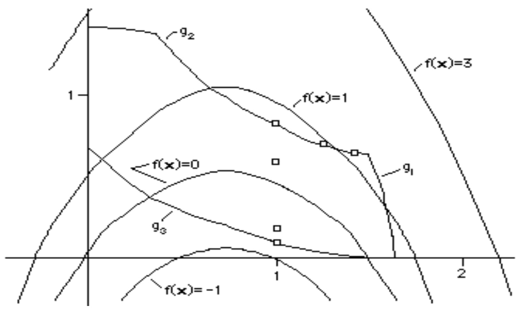

Figure 8. Sequence of six basic solutions during phase 2

If the objective in the example is changed from minimization to maximization and the same approximation is used, the algorithm converges to $\mathbf{x} = (1.458, 0.6)$. This point is a local maximum approximating the rightmost intersection of the boundaries of constraints g_1 and g_2 located at $\mathbf{x} = (1.449, 0.652)$. The global maximum resides at $\mathbf{x} = (0, 1.414)$. The fact that it was not found can be attributed to the nonconvexity of the constraint region. We are assured that a local maximum is a global maximum if and only if a *concave* function is being maximized over a convex constraint region. Although the algorithm found the global minimum to the example, this was only because we were minimizing a convex objective function and the global optimum happened to reside on the boundary of a constraint, g_3, whose feasible region is convex.

Convex Programming Problems

These observations stem directly from the fact that the separable programming method guarantees an approximate global optimum to the original problem only when one is minimizing a convex function (maximizing a concave function) over a convex set. When these conditions hold, the accuracy of the approach is limited only by the coarseness of the piecewise linear approximations that are used. Furthermore, when solving a convex programming problem, we may solve the approximate problem using ordinary linear programming methods without enforcing the adjacency restrictions.

Nonconvex Programming

If the conditions that define a convex program are not present, several outcomes may occur.

1. An approximate global optimum is found (as in the minimization example above).
2. An approximate local optimum is found that is not the global optimum.
3. The solution to the approximate problem may be infeasible with respect to the original problem or be nowhere near a corresponding local or global optimum. These outcomes are due to the fact that an insufficient number of lines segments were chosen for the approximation. In many cases, however, infeasible solutions will be only *slightly* infeasible, and thus present no practical difficulty.

Notwithstanding the possibility of obtaining erroneous results, separable programming methods have proven to be very useful in a variety of practical applications. In addition, it is possible to modify the basic transformations by introducing integer variables and related constraints so that approximate global optima are always obtained, regardless of the convexity of the original problem. An experimental code called MOGG was developed by Falk and Soland (1969) along these lines. Unfortunately, the modified formulation often yields problems that are extremely difficult to solve within acceptable time limits.