Separable Programming

This Hypercard stack was prepared by:
Dennis L. Bricker,
Dept. of Industrial Engineering,
University of Iowa,
Iowa City, Iowa 52242
e-mail: dbricker@icaen.uiowa.edu

author

- Definition of separability
- Piecewise-Linear Optimization
- Restricted Basis Entry rules
- Example
- Refining the Grid

A function $f(x_1, x_2, \cdots x_n)$ is *separable* if it can be written as a sum of terms, each term being a function of a *single* variable:

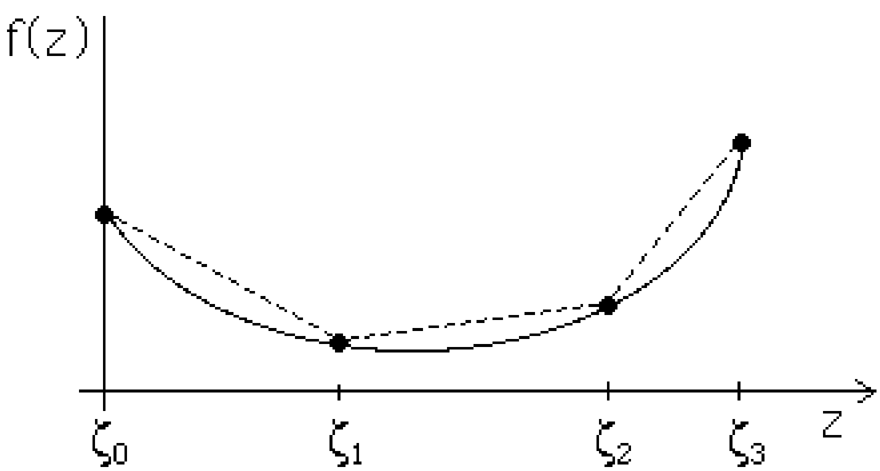
$$f(x_1, x_2, \cdots x_n) = \sum_{i=1}^{n} f_i(x_i)$$

examples

separable	not separable
$\sqrt{x_1} + 2 \ln x_2$	$x_1 x_2 + x_3$
$x_1^2 + 3x_1 + 6x_2 - x_2^2$	$5x_1/x_2 - x_1$

Piecewise-Linear (separable) Programming

We approximate a nonlinear separable function by a piecewise-linear function:



Piecewise-Linear (separable) Programming

There are two ways to formulate the piecewise-linear programming problem as a Linear Programming problem:

- ☞ "LAMBDA" formulation
- ☞ "DELTA" formulation

Piecewise-Linear (separable) Programming

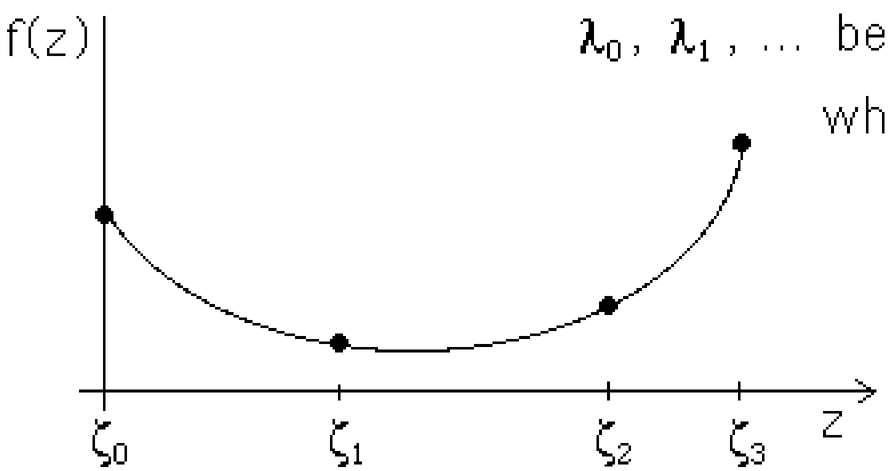
Suppose that f(z) is a ***convex*** function.

Let $\zeta_0, \zeta_1, \ldots$ be specified "grid points", and

$\lambda_0, \lambda_1, \ldots$ be "weights"

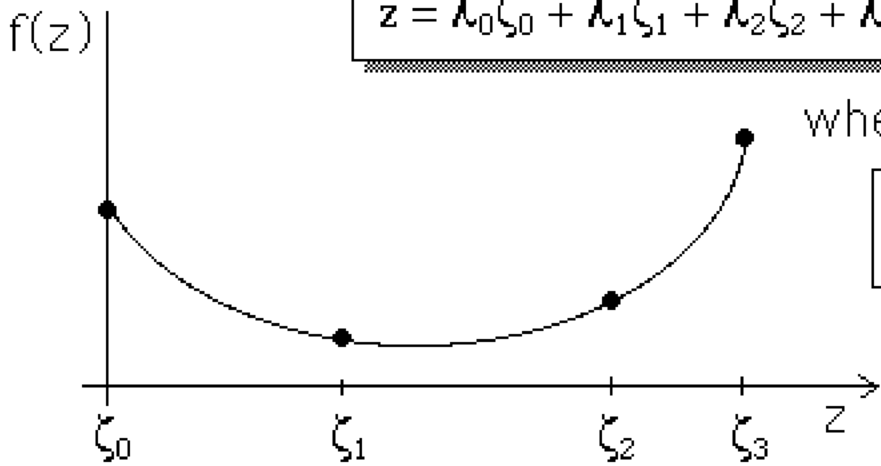
where

$$\sum_i \lambda_i = 1, \lambda_i \geq 0$$



Any value of z in the interval between the left-most and the right-most grid point may be expressed as a **"convex combination"** of the grid points:

$$z = \lambda_0\zeta_0 + \lambda_1\zeta_1 + \lambda_2\zeta_2 + \lambda_3\zeta_3$$



where

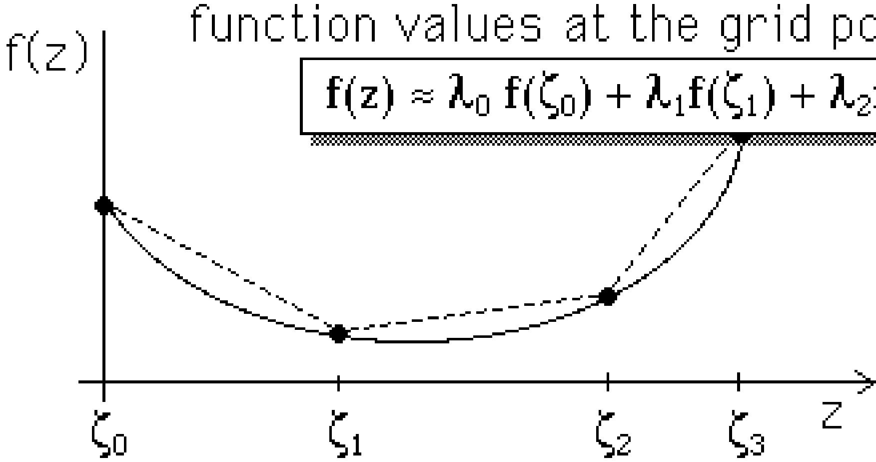
$$\sum_i \lambda_i = 1,\ \lambda_i \geq 0$$

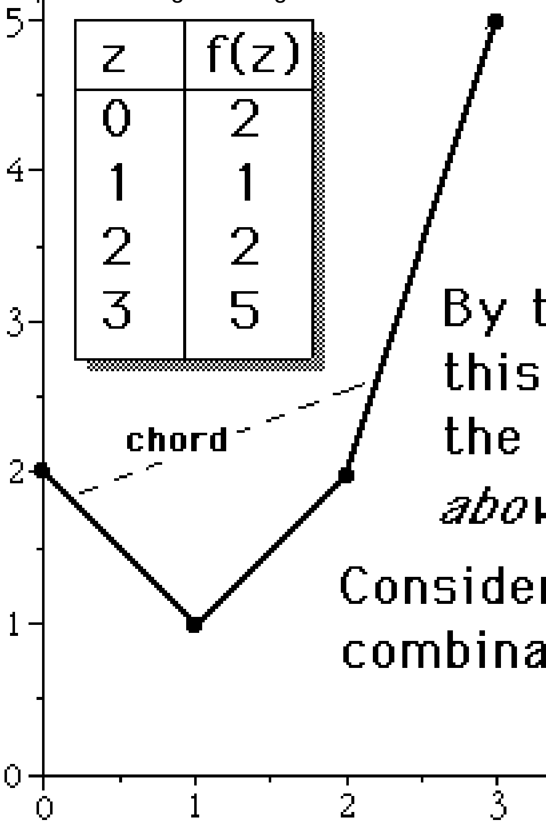
With the same "weights" used in writing the convex combination of the grid points,

$$z = \lambda_0\zeta_0 + \lambda_1\zeta_1 + \lambda_2\zeta_2 + \lambda_3\zeta_3$$

we approximate f(z) as a convex combination of the function values at the grid points

$$f(z) \approx \lambda_0\, f(\zeta_0) + \lambda_1 f(\zeta_1) + \lambda_2 f(\zeta_2) + \lambda_3 f(\zeta_3)$$



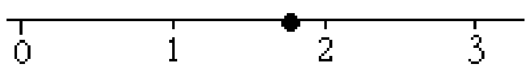


z	f(z)
0	2
1	1
2	2
3	5

Suppose that f(z) is piecewise linear and *convex*...

By the definition of "convex", this means that every chord of the graph of f(z) lies *on* or *above* the graph!

Consider now the various convex combinations of grid points yielding z=1.75.....



A given value of z, e.g., z=1.75, can be represented by several different convex combinations of the grid points:

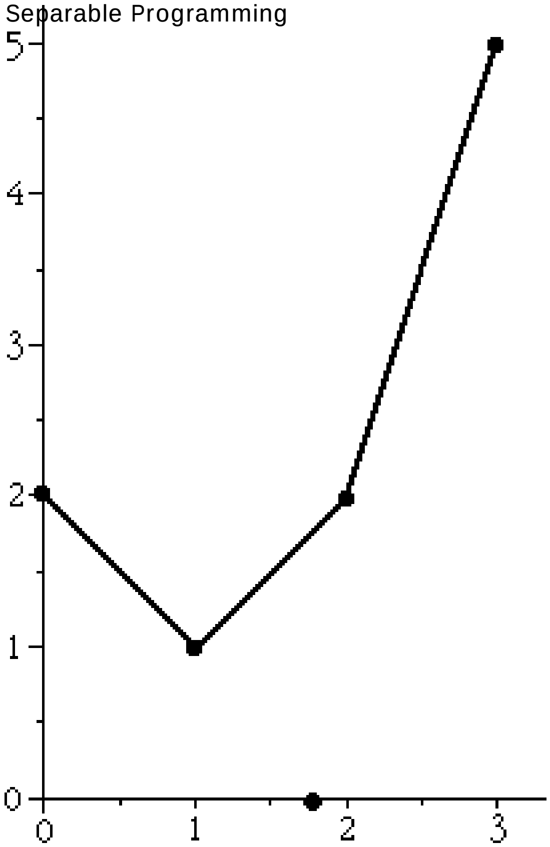
$$1.75 = \frac{5}{12}(0) + \frac{7}{12}(3)$$

$$1.75 = \frac{5}{8}(1) + \frac{3}{8}(3)$$

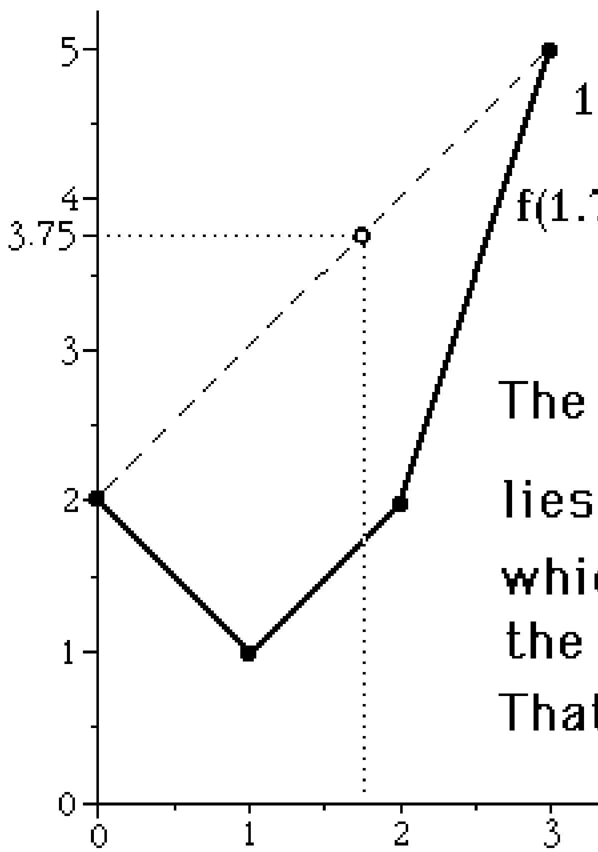
$$1.75 = \frac{1}{2}(1) + \frac{1}{4}(2) + \frac{1}{4}(3)$$

$$1.75 = \frac{1}{4}(1) + \frac{3}{4}(2)$$

etc.



Each set of "weights" in the convex combinations (which yield the same z) when used to weight the function values, will result in a different approximation to f(z).



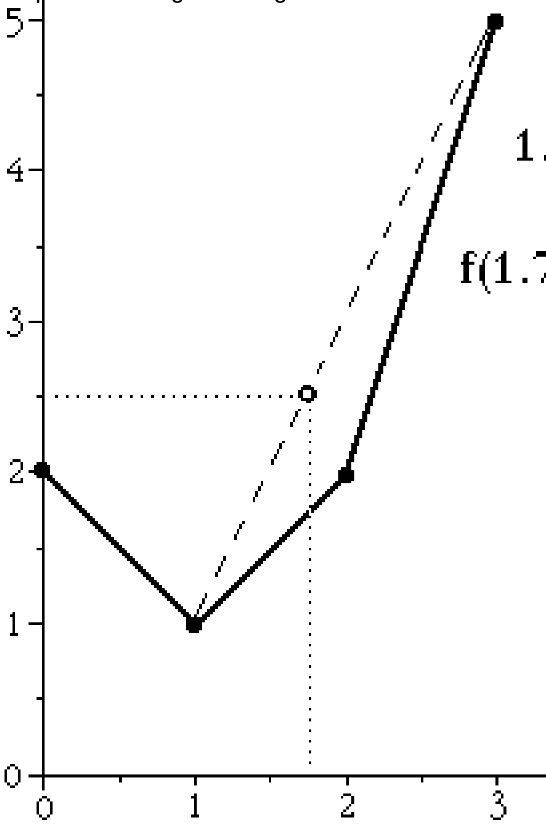
$$1.75 = \frac{5}{12}(0) + \frac{7}{12}(3)$$

$$f(1.75) \approx \frac{5}{12} f(0) + \frac{7}{12} f(3) = \frac{15}{4}$$

The point $\left(\sum_i \lambda_i \zeta_i , \sum_i \lambda_i f(\zeta_i)\right)$

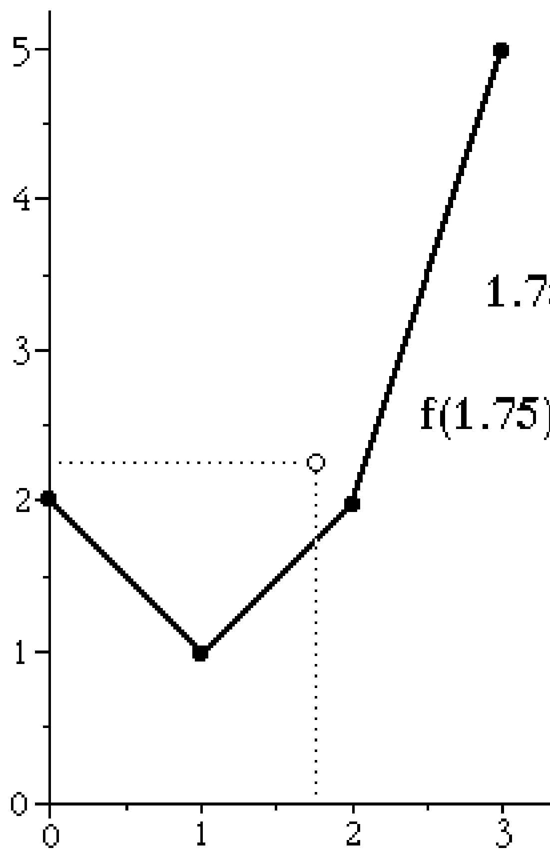
lies on a chord of the graph which is, of course, on or above the graph.

That is, $\sum_i \lambda_i f(\zeta_i)$ is in general an **overestimate** of f(z).



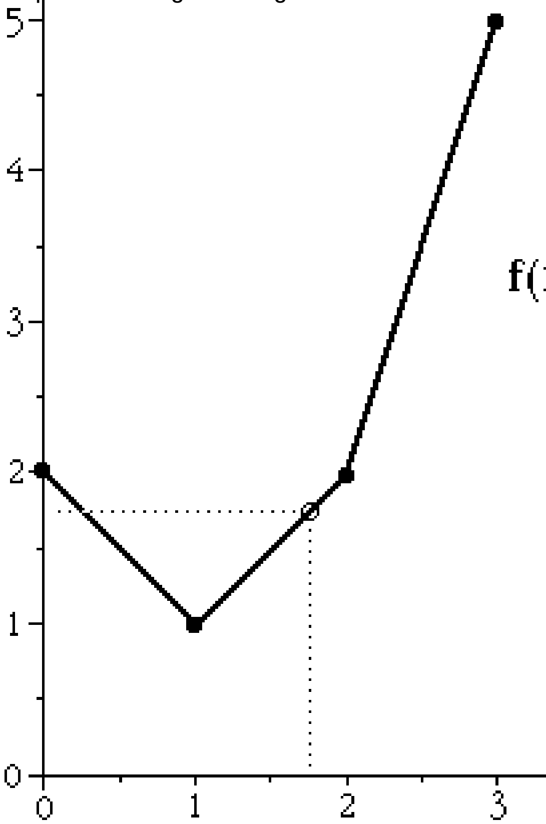
$$1.75 = \frac{5}{8}(1) + \frac{3}{8}(3)$$

$$f(1.75) \approx \frac{5}{8}f(1) + \frac{3}{8}f(3) = \frac{5}{2}$$



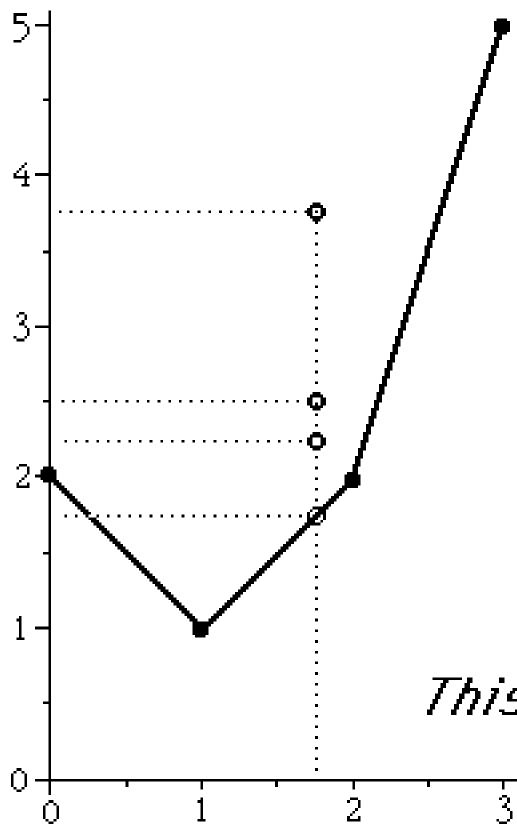
$$1.75 = \frac{1}{2}(1) + \frac{1}{4}(2) + \frac{1}{4}(3)$$

$$f(1.75) \approx \frac{1}{2}f(1) + \frac{1}{4}f(2) + \frac{1}{4}f(3) = \frac{9}{4}$$



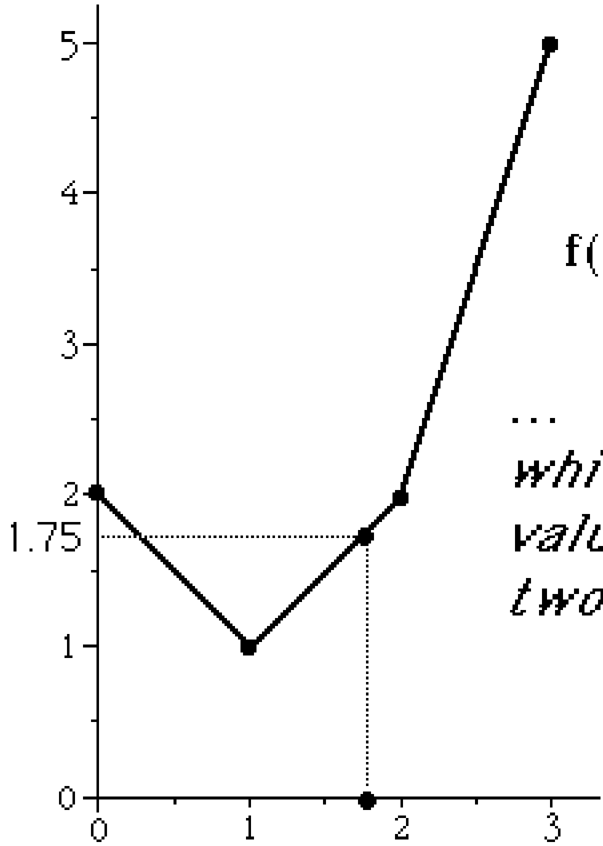
$$1.75 = \frac{1}{4}(1) + \frac{3}{4}(2)$$

$$f(1.75) \approx \frac{1}{4}f(1) + \frac{3}{4}f(2) = \frac{7}{4}$$



Of the various ways to express z as a convex combination of grid points, the way which results in the *minimum* value for an approximation of f(z) is that which assigns positive weights only to the grid points immediately to the left and right of z.

This is the convex combination which best approximates f(z) !



$$1.75 = \frac{1}{4}(1) + \frac{3}{4}(2)$$

$$f(1.75) = \frac{1}{4} f(1) + \frac{3}{4} f(2) = \frac{7}{4}$$

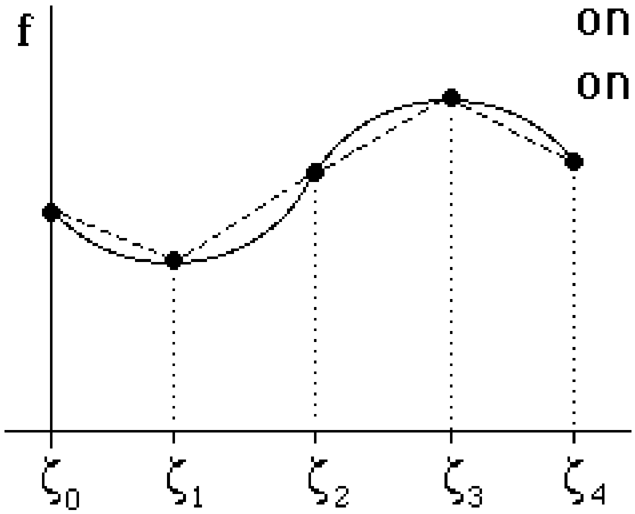
... the convex combination which yields the LOWEST value for f(1.75) uses only two ADJACENT grid points!

When minimizing a *convex* function f(z) by choosing the weights in the convex combination, then,

...at most TWO λ_i's will be positive, and these will be weights of adjacent grid points!

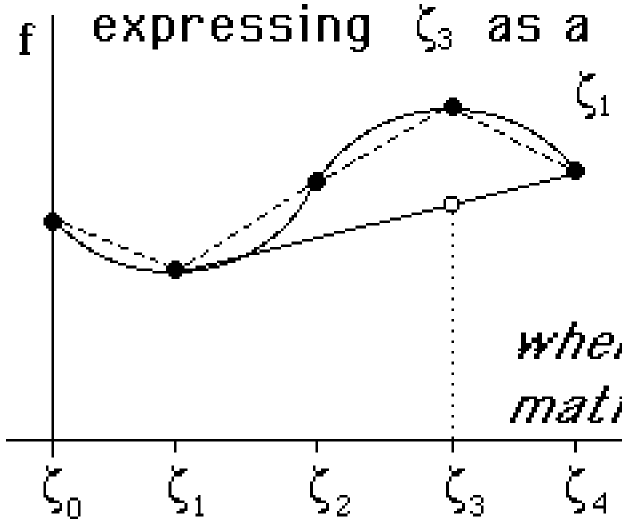
What happens if f(z) is NOT convex?

$$\left(\sum_i \lambda_i \zeta_i \, , \, \sum_i \lambda_i f(\zeta_i)\right)$$



When f(z) is not convex, the chords do not all lie on or above the graph, and one can choose convex combinations of grid points yielding approximations of f(z) which are underestimates of the function.

For example, in this figure, the lowest (and the worst!) estimate of $f(\zeta_3)$ would be obtained by expressing ζ_3 as a convex combination of ζ_1 and ζ_4: $\zeta_3 = \lambda_1 \zeta_1 + \lambda_4 \zeta_4$



with $f(\zeta_3)$ approximated by $\lambda_1 f(\zeta_1) + \lambda_4 f(\zeta_4)$

whereas the "best" approximation is obtained by

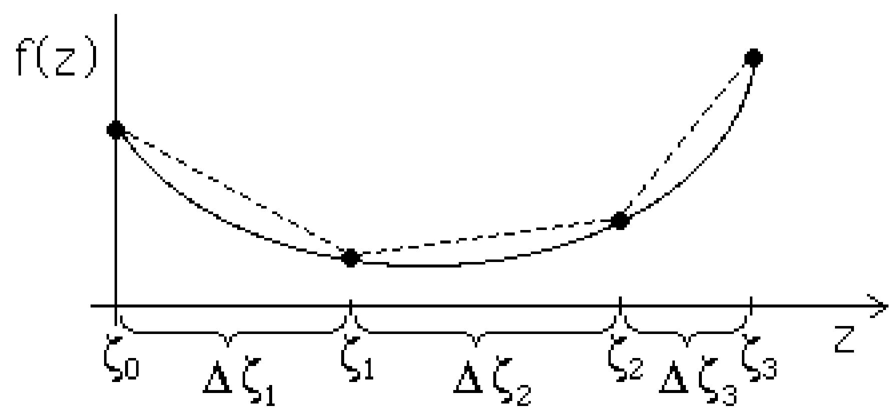
$$\zeta_3 = 0\zeta_1 + 0\zeta_1 + 1\zeta_3 + 0\zeta_4$$

"Delta" form of Separable Programming

In the "lambda" formulation, a special variable (λ) was defined for each grid point. In the "delta" formulation, a special variable (δ) will be defined for each interval between grid points, i.e., for each linear piece.

There are two variations....

"Delta" form of Separable Programming



Define constants:

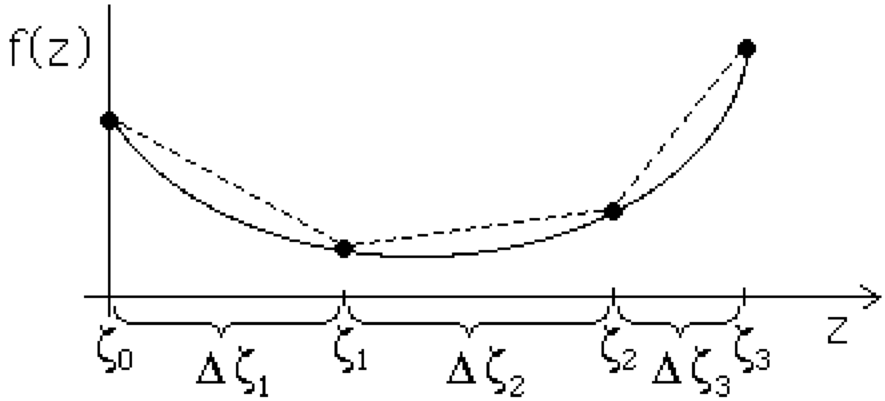
$$\Delta\zeta_i \equiv \zeta_i - \zeta_{i-1}$$
$$\Delta f_i \equiv f(\zeta_i) - f(\zeta_{i-1})$$

Define variables:

$$0 \le \delta_i \le 1 \quad \textit{OR} \quad 0 \le \Delta_i \le \Delta\zeta_i$$

"Delta" form of Separable Programming

variation #1



each variable is bounded between zero and 1.00

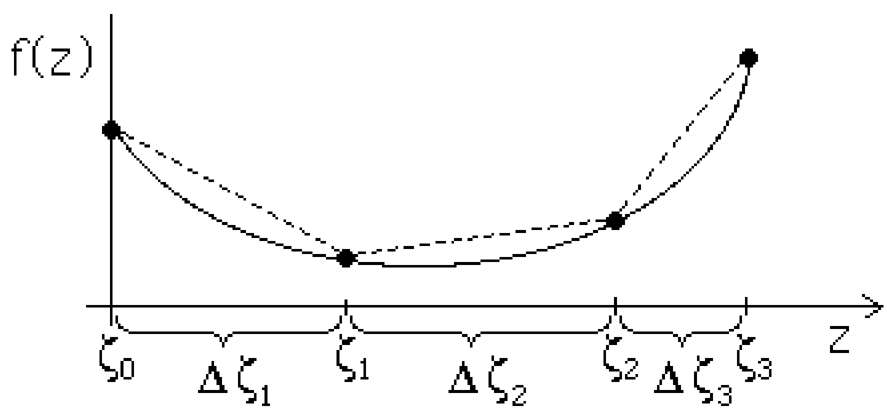
$$z = \zeta_0 + \sum_{i=1}^{p} (\Delta \zeta_i)\delta_i$$

$$f(z) \approx f(\zeta_0) + \sum_{i=1}^{p} (\Delta f_i)\delta_i$$

$$0 \le \delta_p \le \cdots \le \delta_1 \le 1$$

"Delta" form of Separable Programming

variation #2



each variable has an upper bound equal to the length of the interval

$$z = \zeta_0 + \sum_{i=1}^{p} \Delta_i$$

$$f(z) \approx f(\zeta_0) + \sum_{i=1}^{p} \left(\frac{\Delta f_i}{\Delta \zeta_i}\right) \Delta_i$$

$$\Delta_i \equiv (\Delta \zeta_i)\, \delta_i$$

$$0 \le \Delta_i \le \Delta \zeta_i$$

"Delta" form of Separable Programming

In either variation, at most ONE variable is allowed to be at an intermediate value (not a bound), i.e., BASIC when we use UBT (upper bounding technique)

variation #1

$$z = \zeta_0 + \sum_{i=1}^{p} (\Delta\zeta_i)\delta_i$$

$$f(z) \approx f(\zeta_0) + \sum_{i=1}^{p} (\Delta f_i)\delta_i$$

$$0 \le \delta_p \le \cdots \le \delta_1 \le 1$$

variation #2

$$z = \zeta_0 + \sum_{i=1}^{p} \Delta_i$$

$$f(z) \approx f(\zeta_0) + \sum_{i=1}^{p} \left(\frac{\Delta f_i}{\Delta \zeta_i}\right) \Delta_i$$

$$0 \le \Delta_i \le \Delta\zeta_i$$

If we are:

- minimizing a non-convex function

&/or

- optimizing over a nonconvex region
  e.g., $g(x) \le 0$ where g is non-convex,

Then the simplex method will **NOT** yield a basic solution in which

- at most two (adjacent) λ's are basic
  (λ-formulation)
- only one δ is basic
  (δ-formulation)

Restricted Basis Entry Rules

In these cases, a "restricted basis entry" rule may be implemented, which will guarantee that the solution satisfies the desired properties,

- at most 2 λ's are in the basis, in which case they have consecutive indices (λ-formulation)
- at most one δ is in the basis (δ-formulation)

but unfortunately will not guarantee an optimal solution!

Restricted Basis Entry Rules

"Lambda" formulation

Special set: $\{\lambda_{i0}, \lambda_{i1}, \cdots \lambda_{ip}\}$

Constraint

λ_{ij} is positive for at most TWO values of j, in which case they are consecutive indices.

How can we modify the simplex method so as to impose this restriction?

Restricted Basis Entry Rules

"Lambda" formulation

Special set: $\{\lambda_{i0}, \lambda_{i1}, \cdots \lambda_{ip}\}$

Constraint

λ_{ij} is positive for at most TWO values of j, in which case they are consecutive indices.

RBE Rule

If 2 adjacent weights are in the basis, then no other weight from the same set may be considered for basis entry;

if only one weight λ_{ij} is basic, then only $\lambda_{i,j-1}$ & $\lambda_{i,j+1}$ are considered as candidates for basis entry

Restricted Basis Entry Rules

"Lambda" formulation

Special set: $\{\lambda_{i0}, \lambda_{i1}, \cdots \lambda_{ip}\}$

Note that this modification of the simplex method does not guarantee optimality, unless the function being minimized is a convex function!

Restricted Basis Entry Rules

"Delta" formulation

Special set: $\{\delta_{i1}, \delta_{i2}, \cdots \delta_{ip}\}$

Constraint

δ_{ij} is at an intermediate level (neither lower nor upper bound) for at most a single j (i.e., if UBT is used, at most one variable in the set is basic.)

How can we modify the simplex method so as to impose this restriction?

Restricted Basis Entry Rules

"Delta" formulation

Special set: $\{\delta_{i1}, \delta_{i2}, \cdots \delta_{ip}\}$

Constraint

δ_{ij} is at an intermediate level (neither lower nor upper bound) for at most one j (i.e., if UBT is used, at most one variable in the set may be basic.)

RBE Rule

δ_{ij} is not considered for basis entry unless:

- no other variable in the set is basic
- $\delta_{i,j-1}$ is at upper bound
- $\delta_{i,j+1}$ is at lower bound

Restricted Basis Entry Rules

"Delta" formulation

Special set: $\{\delta_{i1}, \delta_{i2}, \cdots \delta_{ip}\}$

RBE Rule

Example

$$\underbrace{1,\ 1,\ 1,\ 1,}_{U}\ \underbrace{3/8}_{B},\ \underbrace{0,\ 0,\ 0,\ 0,\ 0}_{L}$$

no variable may enter the basis

Restricted Basis Entry Rules

"Delta" formulation

Special set: $\{\delta_{i1}, \delta_{i2}, \cdots \delta_{ip}\}$

RBE Rule

Example

considered for basis entry

$$\underbrace{1,\ 1,\ 1,\ 1,\ 1,}_{U}\ \underbrace{0,\ 0,\ 0,\ 0,\ 0,\ 0}_{L}$$

In this case, no variable in the set is in the basis set B; one variable in L and one variable in U may enter B

Example

A company manufactures three products, using three limited resources:

resources	product A	product B	product C	available supply
ingredient #1	1	2	1	1000
ingredient #2	10	4	5	7000
ingredient #3	3	2	1	4000

Because of various factors (e.g., quantity discounts, use of overtime, etc.) the profits per unit decrease as sales increase:

product A

sales	profit ($/unit)
0-40	10
40-100	9
100-150	8
over 150	7

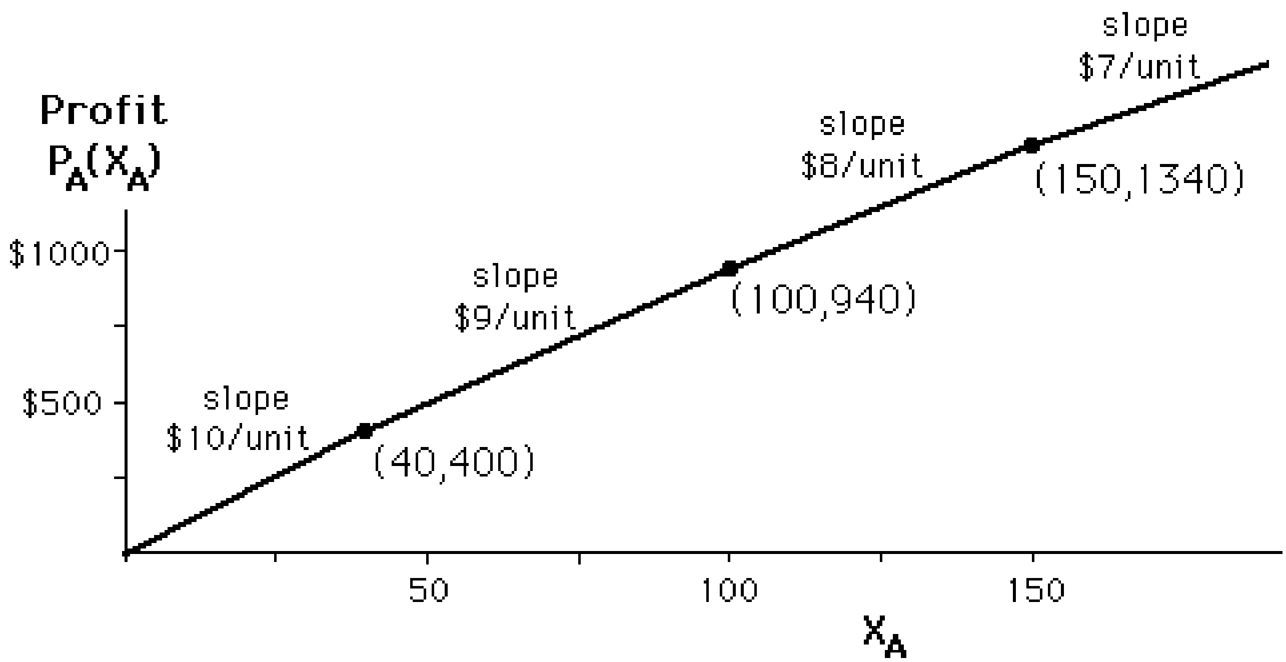
product B

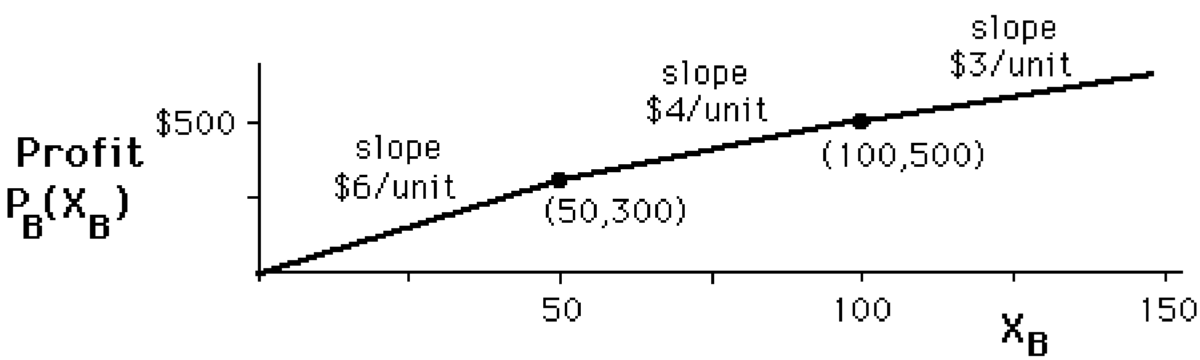
sales	profit ($/unit)
0-50	6
50-100	4
over 100	3

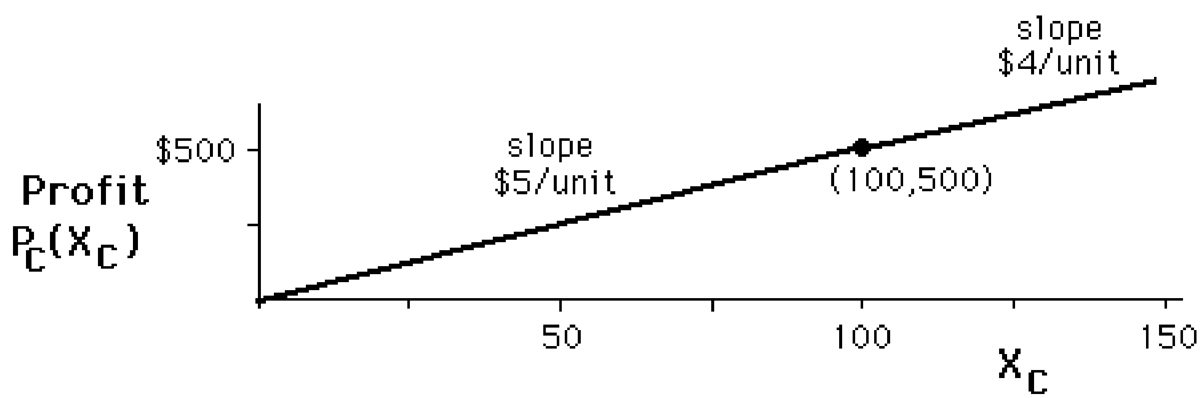
product C

sales	profit ($/unit)
0-100	5
over 100	4

Determine the most profitable mix of products


Profit
$P_A(X_A)$
$1000
$500
slope
$10/unit
slope
$9/unit
slope
$8/unit
slope
$7/unit
(40,400)
(100,940)
(150,1340)
50
100
150
X_A


Profit
$P_B(X_B)$
$500
slope
$6/unit
slope
$4/unit
slope
$3/unit
(50,300)
(100,500)
50
100
150
X_B


Profit
$P_C(X_C)$
$500
slope
$5/unit
slope
$4/unit
(100,500)
50
100
150
X_C

Maximize $p_A(x_A) + p_B(x_B) + p_C(x_C)$
subject to

$$\begin{cases} x_A + 2x_B + x_C \le 1000 \\ 10x_A + 4x_B + 5x_C \le 7000 \\ 3x_A + 2x_B + x_C \le 4000 \end{cases}$$

$$x_A \ge 0,\ x_B \ge 0,\ x_C \ge 0$$

Each profit function p_A, p_B, & p_C,
is piecewise linear.

We can reformulate this as a linear programming problem in two ways:

☞ "delta" formulation
one variable for each interval

☞ "lambda" formulation
one variable for each grid point

©Dennis Bricker, U. of Iowa, 1998

"Delta" formulation

Define

Δ_{A1} = quantity of A produced at $10/unit profit,
Δ_{A2} = quantity of A produced at $9/unit profit,
... etc.

so that

$$p_A(x_A) = 10\Delta_{A1} + 9\Delta_{A2} + 8\Delta_{A3} + 7\Delta_{A4}$$

$$0 \le \Delta_{A1} \le 40$$
$$0 \le \Delta_{A2} \le 60 \quad = 100-40$$
$$0 \le \Delta_{A3} \le 50 \quad = 150-100$$
$$0 \le \Delta_{A4}$$

Since the simplex algorithm will maximize, the optimum will NOT use a positive value for Δ_{A2} unless the more profitable Δ_{A1} has reached its upper limit (40), etc.

Thus, the simplex algorithm will naturally impose the restricted basis entry (RBE) rules.

(these profit functions exhibit "decreasing returns to scale"....)

	Δ_{A1}	Δ_{A2}	Δ_{A3}	Δ_{A4}	Δ_{B1}	Δ_{B2}	Δ_{B3}	Δ_{C1}	Δ_{C2}		
Max	10	9	8	7	6	4	3	5	4		
	1	1	1	1	2	2	2	1	1	≤	1000
	10	10	10	10	4	4	4	5	5	≤	7000
	3	3	3	3	2	2	2	1	1	≤	4000
lower bounds	0	0	0	0	0	0	0	0	0		
upper bounds	40	60	50	∞	50	50	∞	100	∞		

"Lambda" formulation

We require an upper bound (right-most grid point) for each product A, B, and C. Let's arbitrarily use 1000 for each.

Define a weight for each grid point:

$\lambda_{A0} \leftrightarrow 0$

$\lambda_{A1} \leftrightarrow 40$

$\lambda_{A2} \leftrightarrow 100$

$\lambda_{A3} \leftrightarrow 150$

$\lambda_{A4} \leftrightarrow 1000$

"Lambda" formulation

Substitute

$$p_A(x_A) = 0\,\lambda_{A0} + 400\,\lambda_{A1} + 940\,\lambda_{A2} + 1340\,\lambda_{A3} + 6590\,\lambda_{A4}$$

and

$$x_A = 0\,\lambda_{A0} + 40\,\lambda_{A1} + 100\,\lambda_{A2} + 150\,\lambda_{A3} + 1000\,\lambda_{A4}$$

... etc.

"Lambda" formulation

Max	0	400	940	1340	6590	0	300	500	3200	0	500	4100		
	0	40	100	150	1000	0	100	200	2000	0	100	1000	≤	1000
	0	400	1000	1500	10000	0	200	400	4000	0	500	5000	≤	7000
	0	120	300	450	3000	0	100	200	2000	0	100	1000	≤	4000
	1	1	1	1	1								=	1
						1	1	1	1				=	1
										1	1	1	=	1