5 Multivariate Interpolation

**Definition 5.1.** Let \( x_1, x_2, \ldots, x_N \) denote \( N \) distinct points in \( \mathbb{R}^D \), and \( \phi_1, \phi_2, \ldots, \phi_N \) denote \( N \) linearly independent continuous functions \( \mathbb{R}^D \to \mathbb{R} \). The multivariate interpolation problem seeks \( a_1, a_2, \ldots, a_N \in \mathbb{R} \) such that
\[
\forall j = 1, 2, \ldots, N, \quad \sum_{i=1}^{N} a_i \phi_i(x_j) = f(x_j),
\]
where \( f : \mathbb{R}^D \to \mathbb{R} \) is a given function.

**Definition 5.2.** The sites \( x_1, x_2, \ldots, x_N \) of the multivariate interpolation problem are said to be poised with respect to the basis functions \( \phi_1, \phi_2, \ldots, \phi_N \) iff the sample matrix
\[
S = \begin{bmatrix} \phi_1(x_1) & \phi_2(x_1) & \cdots & \phi_N(x_1) \\ \phi_1(x_2) & \phi_2(x_2) & \cdots & \phi_N(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(x_N) & \phi_2(x_N) & \cdots & \phi_N(x_N) \end{bmatrix}
\]
is nonsingular.

**Theorem 5.3.** The multivariate interpolation problem has a unique solution if and only if its sites are poised.

**Example 5.1.** Suppose that the values of a function \( f : \mathbb{R}^2 \to \mathbb{R} \) are known at the sites \((1,0), (-1,0), (0,1), \) and \((0,-1)\). For the basis functions \( 1, x, y, xy \), the sample matrix
\[
S = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & -1 & 0 \end{bmatrix}
\]
is clearly singular, and hence this multivariate interpolation problem does not admit a unique solution.

5.1 Rectangular grids

**Theorem 5.4** (Lagrange formula for rectangular grids). Given two subsets of \( \mathbb{R} \) as \( X = \{x_0, x_1, \ldots, x_m\} \) and \( Y = \{y_0, y_1, \ldots, y_n\} \), the multivariate interpolation problem on the rectangular grid \( X \times Y \) with the set of basis functions
\[
\Phi = \{x^i y^j : i = 0, 1, \ldots, m; j = 0, 1, \ldots, n\}
\]
is solved by the unique solution
\[
p(x, y) = \sum_{i=0}^{m} \sum_{j=0}^{n} f(x_i, y_j) L_i(x) M_j(y),
\]
where \( L_i(x) \) and \( M_j(y) \) are the elementary Lagrange interpolation polynomials defined in (3.9).

**Proof.** Define a blending function
\[
\xi(x, y) = \sum_{i=0}^{m} f(x_i, y) L_i(x)
\]
and apply Theorem 3.4 and Definition 3.8 dimension-by-dimension in a recursive manner.

**Corollary 5.5.** The unique solution in Theorem 5.4 can also be expressed via divided differences as
\[
p(x, y) = \sum_{i=0}^{m} \sum_{j=0}^{n} \pi_i(x) \pi_j(y) |x_0, \ldots, x_i| |y_0, \ldots, y_j| f(x, y),
\]
where \( \pi \) is defined in (3.10), and each divided difference acts on the function \( f \) with the other coordinate fixed.

**Proof.** Theorem 5.4 and Definition 3.11 yield (5.6). □

**Example 5.2.** Write down the unique interpolating polynomial for the following data
\[
\begin{array}{cccc} \hline (x, y) & (1, -1) & (-1, 1) & (1, -1) & (1, 1) \\ \hline f(x, y) & 1 & 5 & -5 & 3 \\ \hline \end{array}
\]

**Proof.** To apply (5.6), we calculate
\[
\begin{align*}
[-1, 1]_x f(-1, y) &= \frac{-5 - 1}{1 + 1} = -3, \\
[-1, 1]_y f(-1, y) &= \frac{5 - 1}{1 + 1} = 2, \\
[-1, 1]_y f(x, 1) &= \frac{-5 - 1}{1 + 1} = -3, \\
[-1, 1]_y f(x, 1) &= \frac{5 - 1}{1 + 1} = 2.
\end{align*}
\]
It follows that the unique interpolating polynomial is
\[
p(x, y) = 1 - 3(x + 1) + 2(y + 1) + (x + 1)(y + 1)
\]
\[= 1 - 2x + 3y + xy.\]

**Lemma 5.6.** For \( k, \ell \in \mathbb{N}^+ \), define
\[
(\Delta_x^k \Delta_y^\ell) f(x, y) = \Delta_y^\ell (\Delta_x^k f(x, y)).
\]
Then the two difference operators \( \Delta_x^k \) and \( \Delta_y^\ell \) commute,
\[
\Delta_x^k \Delta_y^\ell f(x, y) = \Delta_y^\ell \Delta_x^k f(x, y).
\]

**Proof.** (5.8) follows from (5.7) and Definition 3.19. □

**Corollary 5.7.** Consider a rectangular grid with uniform spacing along each dimension,
\[
\forall i = 0, 1, \ldots, m, \quad x_i = x_0 + ih_x; \\
\forall j = 0, 1, \ldots, n, \quad y_j = y_0 + jh_y.
\]
The unique solution in Theorem 5.4 can also be expressed via forward differences as
\[
p(x_0 + sh_x, y_0 + th_y) = \sum_{i=0}^{m} \sum_{j=0}^{n} \binom{s}{i} \binom{t}{j} \Delta_x^i \Delta_y^j f(x_0, y_0),
\]
where \( \binom{s}{i} \) and \( \binom{t}{j} \) are defined in (3.35).
Proof. (5.10) follows from Theorem 3.23, Lemma 5.6, and the structure of rectangular grids.

Remark 5.3. One cannot form a complete symmetric polynomial of degree \(nm\) from the basis functions (5.3). In fact, the degree of a complete symmetric polynomial can only be \(0, 1, \ldots, \min(n, m)\). Hence the best order of accuracy of the interpolation polynomial (5.4) is \(\min(n, m) + 1\), which highlights a redundancy of interpolation sites and basis functions. This motivates multivariate interpolation on triangular grids.

5.2 Triangular grids

Lemma 5.8. Given \(n\) distinct points \(x_i \in \mathbb{R}\) and another \(n\) distinct points \(y_j \in \mathbb{R}\). For the set of triangular interpolation sites

\[
T^n = \{(x_i, y_j) : i, j \geq 0, i + j \leq n\}
\]

and the set of triangular basis functions

\[
\Phi_\Delta = \{1, x, y, x^2, xy, y^2, \ldots, x^n, x^{n-1}y, \ldots, xy^{n-1}, y^n\},
\]

the corresponding sample matrix \(S_\Delta\) satisfies

\[
\det S_\Delta = C\psi_n(x)\psi_n(y)
\]

where \(C\) is a nonzero constant and

\[
\psi_n(x) = \psi(x_0, x_1, \ldots, x_n) = \prod_{i>j}(x_i - x_j)^{n+1-i}.
\]

Proof. The basis functions in (5.12) are the terms in complete symmetric polynomials in two variables of degree \(0, 1, \ldots, n\). For fixed \(i, k\) with \(i > k\), the replacement of the site \((x_i, y_j)\) with \((x_k, y_j)\) makes the sample matrix singular; the number of this type of replacements is \(n + 1 - i\). Repeat this argument for all \(i = 1, 2, \ldots, n\), and we know that \(\psi_n(x)\) is a factor of \(\det S_\Delta\). For a single fixed \(i\), there are \(i\) indices less than \(i\), contributing to a total degree of \(1 + 2 + \cdots + i = \frac{i(i+1)}{2}\). By an easy induction, one can show that the total degree of \(\psi_n(x)\) in the variables \(x_0, x_1, \ldots, x_n\) is

\[
\sum_{i=1}^{n} \frac{i(i+1)}{2} = \frac{n(n+1)(n+2)}{6}.
\]

Similarly, \(\det S_\Delta\) must contain a factor of \(\psi_n(y)\), of which the total degree is also (5.15). By Definitions 5.2 and 0.83, each term of \(\det S_\Delta\) is a product of the basis functions in (5.12) evaluated at some \(x_i, y_j\). Hence the total degree of \(\det S_\Delta\) in the variables \(x_0, x_1, \ldots, x_n\) and \(y_0, y_1, \ldots, y_n\) is

\[
\sum_{i=1}^{n} \frac{i(i+1)}{2} = \frac{n(n+1)(n+2)}{3}.
\]

The proof is completed by the fact that two complete polynomials of the same degree must be related together by scalar multiplication.

Theorem 5.9. For distinct \(x_i\)'s and distinct \(y_j\)'s, \(T_\Delta^n\) in (5.11) is poised and a unique polynomial of the form

\[
p_m(x, y) = \sum_{k=0}^{n} \sum_{r=0}^{k} c_{r,k-r}x^ry^{k-r}
\]

interpolates any function \(f\) whose domain includes \(T_\Delta^n\).

Proof. This follows from Lemma 5.8 and Theorem 5.4.

Remark 5.4. The set of sites \(T_\Delta^n\) has a triangular formulation for \(x_j = y_j = j\). However, the sites may lie in a formation that bears no resemblance to a triangle, e.g.

\[
T_\Delta = \{(0, 0), (-1, 1), (0, 1), (-1, -1), (0, -1), (1, -1)\}.
\]

Theorem 5.10. For any scalar function \(f\) whose domain includes \(T_\Delta^n\), we have,

\[
\forall m = 0, 1, \ldots, n, \quad f(x, y) = p_m(x, y) + r_m(x, y),
\]

where the polynomial \(p_m(x, y)\) interpolates \(f(x, y)\) on \(T_\Delta^n\) with \(r_m(x, y)\) being the remainder,

\[
P_m(x, y) = \begin{cases} [x_0][y_0]f = f(x_0, y_0), & m = 0; \\ [p_{m-1}(x, y) + q_m(x, y)], & m > 0, \\ \end{cases}
\]

\[
q_m(x, y) = \sum_{k=0}^{m} \pi_k(x)\pi_{m-k}(y)[x_0, \ldots, x_k][y_0, \ldots, y_{m-k}]f,
\]

\[
r_m(x, y) = \sum_{k=0}^{m} \pi_{k+1}(x)\pi_{m-k}(y)[x_0, \ldots, x_k][y_0, \ldots, y_{m-k}]f + \pi_{m+1}(y)[x][y_0, \ldots, y_m]f.
\]

Proof. The polynomial \(p_m(x, y)\) clearly interpolates \(f(x, y)\) on \(T_\Delta^n\) because, for each \((x_i, y_j) \in T_\Delta^n\), all the \(m + 2\) terms of \(r_m(x, y)\) in (5.19c) are identically zero. The total degree of \(p_m(x, y)\) is \(m\) while that of \(r_m(x, y)\) is \(m + 1\). It is easily verified that

\[
f(x, y) = f(x_0, y_0) + (x - x_0)[x_0][y_0]f + (y - y_0)[x][y_0]f.
\]

Hence (5.18) and (5.19) hold for the induction basis \(m = 0\). Assume that (5.18) holds for \(m \geq 0\). For the inductive step, we define

\[
S_1 = \sum_{k=0}^{m} \pi_{k+1}(x)\pi_{m-k}(y)[x_0, \ldots, x_{k+1}][y_0, \ldots, y_{m-k}]f,
\]

\[
S_2 = \sum_{k=0}^{m} \pi_{k+2}(x)\pi_{m-k}(y)[x_0, \ldots, x_{k+1}][y_0, \ldots, y_{m-k}]f,
\]

\[
T_1 = \pi_{m+1}(y)[x_0][y_0, \ldots, y_{m+1}]f,
\]

\[
T_2 = \pi_1(x)\pi_{m+1}(y)[x, x_0][y_0, \ldots, y_{m+1}]f,
\]

\[
T_3 = \pi_{m+2}(y)[x][y_0, \ldots, y_{m+1}]f.
\]

Using (3.17), it is not difficult to prove

\[
S_1 + T_1 = q_{m+1}(x, y) = p_{m+1}(x, y) - p_m(x, y),
\]

\[
S_2 + T_2 + T_3 = r_{m+1}(x, y),
\]

\[
r_m(x, y) = r_{m+1}(x, y) + q_{m+1}(x, y).
\]

Hence we have

\[
r_{m+1}(x, y) + p_{m+1}(x, y) = r_m(x, y) + p_m(x, y) = f(x, y),
\]

which completes the inductive proof.
Corollary 5.11. The interpolating polynomial on $\mathcal{T}_n^\alpha$ in Theorem 5.10 can also be expressed as
\[
p(x, y) = \sum_{m=0}^{n} \sum_{k=0}^{m} \pi_k(x) \pi_{m-k}(y) [x_0, \ldots, x_k][y_0, \ldots, y_{m-k}] f.
\] (5.20)

Proof. This follows directly from (5.19a) and (5.19b). □

The rest of this section concerns a special type of triangular grids.

Corollary 5.12. For the principal lattice
\[
\mathcal{P}^n_\Delta = \{(i, j) : i, j \in \mathbb{N}^+, i + j \leq n\},
\] (5.21)
the unique interpolating polynomial in Theorem 5.10 can be expressed as
\[
p(x, y) = \sum_{m=0}^{n} \sum_{k=0}^{m} \binom{x}{k} \binom{y}{m-k} \Delta_x^k \Delta_y^{m-k} f(0, 0).
\] (5.22)

Proof. This follows from Corollary 5.11 and Theorem 3.22. □

Remark 5.5. Corollaries 5.11 and 5.12 are complements of Corollaries 5.5 and 5.7.

Theorem 5.13 (Lagrange formula for principal lattices). The unique interpolating polynomial on the principle lattice (5.21) can be expressed as
\[
p_n(x, y) = \sum_{i, j} f(i, j) L_{i,j}(x, y),
\] (5.23)
where $(i, j) \in \mathcal{P}^n_\Delta$ and the fundamental polynomial is
\[
L_{i,j}(x, y) = \prod_{s=0}^{i-1} \frac{x-s}{i-s} \prod_{j=0}^{j-1} \frac{y-s}{j-s} \prod_{s=i, j+1}^{n} \frac{x+y-s}{i+j-s} = \binom{x}{i} \binom{y}{j} \binom{n-x-y}{n-i-j}.
\] (5.24)

Proof. Clearly we only need to show
\[
L_{i,j}(x, y) = \begin{cases} 
1 & \text{if } x = i, y = j; \\
0 & \text{otherwise}
\end{cases}
\]
It is trivial to verify $L_{i,j}(i, j) = 1$ in (5.24). As for the second clause, the following families of straight lines
\[
\bullet x = 0, 1, \ldots, i-1, \\
y = 0, 1, \ldots, j-1, \\
x + y = i + j + 1, i + j + 2, \ldots, n,
\]
contains all sites of $\mathcal{P}^n_\Delta \setminus \{(i, j)\}$, hence at least one factor in (5.24) is zero at any site. □

Example 5.6. The principle lattice $\mathcal{P}^2_\Delta$ contains six sites and the corresponding interpolating polynomial is
\[
p_2(x, y) = \frac{1}{2} (x+y-1)(x+y-2)f(0, 0) + xy f(1, 1) - x(x+y-2)f(1, 0) + \frac{1}{2} x(x+1)f(2, 0) - y(x+y-2)f(0, 1) + \frac{1}{2} y(y-1)f(0, 2)
\]
The evaluation of $p_2(x, y)$ at the centroid of the triangle with vertices $(0, 0)$, $(2, 0)$, and $(0, 2)$ yields
\[
p_2 \left( \frac{2}{3}, \frac{2}{3} \right) = \frac{1}{3} (4\alpha - \beta),
\]
where $\beta$ is the mean of the values of $f$ on the triangle vertices and $\alpha$ that on the other sites.

Theorem 5.14 (Neville-Aitken formula for principle lattices). Define $p^0_{k}[i,j] = f(i, j)$ and denote by $p^k_{k}[i,j](x, y)$ the unique interpolating polynomial of total degree $k$ for the function $f(x, y)$ on the principle lattice
\[
p^k_{k}[i,j] = \{(i + r, j + s) : r, s \geq 0, r + s \leq k\}.
\] (5.25)

Then, for $k \geq 0$ and $i, j \geq 0$, we have
\[
p^k_{k+1}[i,j](x, y) = \frac{i + j + k + 1 - x - y}{k + 1} p^k_{k}[i,j](x, y) + \frac{x - i}{k + 1} p^{k+1}_{k+1}[i,j](x, y) + \frac{y - j}{k + 1} p^{k+1}_{k+1}[i,j+1](x, y).
\] (5.26)

Proof. The induction basis $k = 0$ clearly holds because $\mathcal{P}^0_0 = \{(i, j)\}$. Suppose $p^k_{k}[i,j](x, y)$ interpolates $f(x, y)$ on $\mathcal{P}^{k+1}_{k+1}$ for all $k \geq 0$ and all $i, j \in \mathbb{N}$. By (5.25), we have
\[
\mathcal{I}_p := \mathcal{P}^{[i,j]}_k \cap \mathcal{P}^{[i+1,j]}_{k+1} \cap \mathcal{P}^{[i,j+1]}_{k+1} = \{(i + r, j + s) : r, s \geq 1, r + s \leq k\}.
\]
The induction hypothesis implies that, $\forall (\ell, m) \in \mathcal{I}_p$,
\[
p^k_{k+1}[i,j](\ell, m) = p^{k+1}_{k+1}[i,j](\ell, m) = p^{k+1}_{k+1}[i,j+1](\ell, m) = f(\ell, m),
\]
which, together with (5.26), yields
\[
\forall (\ell, m) \in \mathcal{I}_p, \quad p^k_{k+1}[i,j](\ell, m) = f(\ell, m).
\]
Similarly, we have
\[
\forall m-j = 1, 2, \ldots, k, \quad p^k_{k}[i,j](i, m) = p^{k+1}_{k}[i,j](i, m) = f(i, m), \\
\forall \ell - i = 1, 2, \ldots, k, \quad p^k_{k}[i,j](\ell, j) = p^{k+1}_{k}[i,j+1](\ell, j) = f(\ell, j).
\]
It follows from the above two equations and (5.26) that
\[
\ell = i \Rightarrow p^k_{k+1}[i,j](\ell, y) = p^{k}_{k}[i,j](\ell, y) = f(\ell, y); \\
m = j \Rightarrow p^k_{k+1}[i,j](x, m) = p^{k}_{k}[i,j](x, m) = f(x, m).
\]
Hence $p^k_{k+1}[i,j](x, y)$ also interpolates $f(x, y)$ on $\mathcal{P}^{[i,j]}_{k+1} \setminus \mathcal{I}_p$ as any site in it satisfies $\ell = i$ or $m = j$. In summary, $p^k_{k}[i,j](x, y)$ interpolates $f(x, y)$ on $\mathcal{P}^{[i,j]}_k$.

The total degree of $p^k_{k}[i,j](x, y)$ being $k$ can also be proved by an easy induction. □

Example 5.7. Use Theorem (5.26) to obtain the last equation in Example 5.6.
5.3 Exact Cover Problem

Definition 5.15 (Exact cover problem). The exact cover problem is a decision problem to find an exact cover. Given a set $S$ and another set where each element is a subset to $S$. Select a set of subsets such that every element in $S$ exist in exactly one of the selected sets. This selection of sets is said to be a cover of the set $S$.

Example 5.8.

\[
\begin{pmatrix}
0 & 1 & 1 \\
1 & 1 & 0 \\
1 & 0 & 0
\end{pmatrix}
\] (5.27)

$S = (1, 1, 1)$, the given subsets are $(0, 1, 1), (1, 1, 0), (1, 0, 0)$. So we choose $1^{st}, 3^{rd}$ subsets as the exact cover of $S$, cover set is $\{1, 3\}$.

Algorithm 5.16 (Algorithm X). Algorithm X is a nondeterministic, recursive algorithm that uses depth-first search for backtracking. It was created by Donald Knuth and can be used to find all solutions to the exact cover problem.

Input: $S$ is a given exact cover problem matrix
Preconditions: $S_{i,j} = 0, 1$
Output: $K$ is the set of row numbers.
Postconditions: every element in $S$ exist in exactly one of the selected sets.

1 if $S$ has no columns then
2 return $K$
3 end
4 if $S$ has a zero column or some empty column’s name is not deleted then
5 Back to upper level
6 end
7 Choose a column $c$, a row $r$ such that $S_{r,c} = 1$
8 $K \leftarrow K \cup r$
9 for Each column $j$ $S_{r,j} = 1$ do
10 for Each row $i$ $S_{i,j} = 1$ do
11 Delete row $i$ from $S$
12 end
13 Delete column $j$ from $S$
14 end
15 AlgorithmX($S$)

Remark 5.9. The nondeterministic choice of row $r$ means that the algorithm essentially clones itself into independent subalgorithms; each subalgorithm inherits the current matrix, but reduces it with respect to a different row $r$.

If column $c$ is entirely zero or column $c$ is not deleted at the end, there are no subalgorithms and the process terminates unsuccessfully.

The subalgorithms form a search tree, with the original problem at the root and with level $k$ containing each subalgorithm that corresponds to rows that chosen in $k-1$ level. Backtracking is the process of traversing the tree in preorder, depth-first.

Example 5.10.

\[
\begin{pmatrix}
A & B & C & D & E & F & G \\
1 & 0 & 0 & 1 & 0 & 1 & 1 \\
2 & 1 & 0 & 0 & 1 & 0 & 0 \\
3 & 0 & 1 & 1 & 0 & 0 & 1 \\
4 & 1 & 0 & 0 & 1 & 0 & 0 \\
5 & 0 & 1 & 0 & 0 & 0 & 1 \\
6 & 0 & 0 & 0 & 1 & 1 & 0
\end{pmatrix}
\] (5.28)

An exact cover problem is given above. $A \sim G$ is column-name of matrix $S$.

Level 0:original problem
- The matrix is not empty, so the algorithm proceeds.
- The lowest number of 1s in any column is two. Choosing one of these column helps to accelerate algorithm. Column A is the first column with two 1s and thus is selected (deterministically):
  - Rows 2 and 4 each have a 1 in column A and thus are selected (nondeterministically).

The algorithm moves to the first branch at level 1. Level 1:select row 2
- Row 2 is included in the partial solution.
- Row 2 has a 1 in columns A, D, and G.
- Delete rows 2, 4, 5, 6. Delete columns A, D, G.
- Select row 2 in Column B of the current matrix (row 3 of the original matrix).

\[
\begin{pmatrix}
B & C & E & F \\
1 & 0 & 1 & 1 \\
3 & 1 & 1 & 0
\end{pmatrix}
\] (5.29)

The algorithm moves to the only branch at level 2. Level 2:select row 3
- Row 3 is included in the partial solution.
- Row 3 has a 1 in columns B, C, and F.
- Delete rows 1, 3. Delete columns B, C, F.
- Column E is not deleted at the end. The process terminates unsuccessfully.

The algorithm to the next branch at level 1. Level 1:select row 4
- Row 4 is included in the partial solution.
- Row 4 has a 1 in columns A, D.
- Delete rows 2, 4, 6. Delete columns A, D.
- Select row 2 in Column B of the current matrix, row 3 of the original matrix.
\[
\begin{pmatrix}
B & C & E & F & G \\
1 & 0 & 1 & 1 & 1 \\
3 & 1 & 1 & 0 & 1 \\
5 & 1 & 0 & 0 & 0 \\
\end{pmatrix}
\]
(5.30)

The algorithm moves to another branch at level 2.
Level 2: select row 3
- Row 3 is included in the partial solution.
- Row 3 has a 1 in columns B, C, and F.
- Delete rows 1, 3, 5. Delete columns B, C, F.
- Column E, G is not deleted at the end. The process terminates unsuccessfully.

The algorithm moves to next branch at level 2.
Level 2: select row 5
- Row 5 is included in the partial solution.
- Row 5 has a 1 in columns B, G.
- Delete rows 3, 5. Delete columns B, G.

\[
\begin{pmatrix}
C & E & F \\
1 & 1 & 1 \\
\end{pmatrix}
\]
(5.31)

Then choose row 1. Then set \(K\) of this exact cover problem is \(\{1, 4, 5\}\).

**Algorithm 5.17** (Dancing Links). **Dancing Links**, DLX for short, is the implementent of Algorithm X by Donald Knuth. The algorithm is based on a data structure, doubly linked list. \(x\) is a given node. L[x] points to the left node of \(x\), R[x] points to the right node of \(x\).

We can easily remove and insert node by modifying pointers.

\[
\begin{align*}
\text{L}[\text{R}[x]] & \leftarrow \text{L}[x], \text{R}[\text{L}[x]] \leftarrow \text{R}[x] \\
\text{L}[\text{R}[x]] & \leftarrow x, \text{R}[\text{L}[x]] \leftarrow x 
\end{align*}
\]
(5.32)

In equation 5.33, notice that R[x], L[x] don’t exist at first. In this equation, they point to the nodes you want to be on the both sides of \(x\).

For DLX, each data object \(x\) have pointers L[x], R[x], U[x], D[x] which link to other cell with an occupying 1 to the left, right, up and down. C[x] points to the column object which is shown as \(A \sim G\) in matrix. S[x] represents the column size which is the number of data objects that are currently linked together from the column object.

```python
Input: h is the root column object,
l is the current level(depth),
s is the solution with a list of data objects.
1 if R[h] = h then
2 print_solution(s)
3 return
4 end
5 c ← choose_column_object(h)
6 r ← D[c]
7 while r ≠ c do
8 \(s ← s + [r]\)
9 \(j ← R[r]\)
10 while j ≠ r do
11 \(\text{cover(C[j])}\)
12 \(j ← R[j]\)
13 \//Solving unsuccessful, pop data then go other branch.
14 r ← s_k
15 c ← C[r]
16 \(j ← L[r]\)
17 while j ≠ r do
18 \(\text{uncover(C[j])}\)
19 \(j ← L[j]\)
20 end
21 uncover(c)
22 return
23 end

Input: c is column object
1 cover(c):
2 \(L[R[c]] \leftarrow L[c]\)
3 \(R[L[c]] \leftarrow R[c]\)
4 \(i ← D[c]\)
5 while i ≠ c do
6 \(j ← R[i]\)
7 \while j ≠ i do
8 \(U[D[j]] \leftarrow U[j]\)
9 \(D[U[j]] \leftarrow D[j]\)
10 \(S[C[j]] \leftarrow S[C[j]] - 1\)
11 \(j ← R[j]\)
12 end
13 i ← D[i]
14 end
```
Example 5.11. Here give a example to show how the link change when we solve the exact cover problem in example refexam:algorithmx.

Select row 2.

Cover column A and row 2, 4.

$$(
\begin{array}{cccccc}
B & C & D & E & F & G \\
1 & 0 & 1 & 0 & 1 & 1 \\
3 & 1 & 1 & 0 & 0 & 1 \\
5 & 1 & 0 & 0 & 0 & 1 \\
6 & 0 & 0 & 1 & 1 & 0 \\
\end{array}
)$$

Cover column D, G and row 5, 6. The following solving process is as same as that in Algorithm X.

\begin{verbatim}
Input: c is column object
1 uncover(c):
  2 i ← D[c]
  3 while i ≠ c do
  4    j ← L[i]
  5     while j ≠ i do
  6         S[C[j]] ← S[C[j]] - 1
  7         U[D[j]] ← U[j]
  8         D[U[j]] ← D[j]
  9     j ← R[j]
10 end
11 i ← U[i]
end
L[R[c]] ← c
R[L[c]] ← c
\end{verbatim}

Definition 5.18 (Sudoku puzzle). Sudoku puzzle is a grid of cells divided into 9 rows and 9 columns. One cell can only contain a single integer between 1 and 9. The grid is also divided into 9 boxes where a box consist of 3 rows and 3 columns. The grid starts out with an arbitrary number of given clue. The goal of the puzzle is to fill the remaining cells in the grid with integers such that every integer appears once in every row, column, box.

If Sudoku puzzle have a solution, it must satisfied the follow condition. Define $G$ as the grid, $G_{i,j}$ as the integer in the cell at $(i,j)$ of grid, $R_i$ as the $i^{th}$ row, $C_j$ as the $j^{th}$ column, $B_p$ as the $p^{th}$ box.

$G_{i,j} = 1, \ldots, 9 \quad (5.34)$
$R_i = \{G_{i,j} : j = 1, \ldots, 9; G_{i,j} ≠ G_{i,j'}, \forall j, j' \} \quad (5.35)$
$C_j = \{G_{i,j} : i = 1, \ldots, 9; G_{i,j} ≠ G_{i',j} ; \forall i, i' \} \quad (5.36)$
$B_p = \{G_{i,j} : i = 3\lfloor \frac{p}{3} \rfloor + 1, \ldots, 3\lfloor \frac{p}{3} \rfloor + 1 \}; \quad (5.37)$

And Sudoku puzzle can also be defined:

$P_{i,j} = \begin{cases} 1, \ldots, 9 & \text{if } P_{i,j} \text{ is given} \\ 1, \ldots, 9 & \text{if } P_{i,j} \text{ is not given} \end{cases} \quad (5.39)$

Example 5.12 (Reducing Sudoku puzzle). Since there are 81 cells, there are 81 columns required for the cell constraint. Since there are 9 rows, 9 columns, 9 boxes in $P$, each row, column, box have 9 cells and each cell have 9 cases of integers, there are $3 \times 81$ columns required for the row, columns, box constraint.

Since each cell have 9 cases of integers, $M$ will have space for its $9 \times 81 = 729$ rows if we don’t have any clues.

For the elment $P_{1,1} = 1$, we can have constraint:

\begin{array}{cccccc}
Cell & Row & Column & Box \\
1 & \ldots & 0 & 1 & \ldots & 0 \\
\end{array}
Algorithm 5.19 (Transfer operator). We can define a transfer operator:

\[ \mathcal{I} : P \rightarrow S \]  

(5.40)

\( P \) is Sudoku puzzle we have defined. \( S \) is a binary matrix of \( \{0,1\} \) which can view as the matrix of exact cover problem.

If \( P_{i,j} \) is given, \( P_{i,j} \) have only one case of integers, otherwise, \( P_{i,j} \) have 9 cases of integers.

**Input:** \( P \) is Sudoku puzzle  
**Output:** \( S \) is a binary matrix of \( \{0,1\} \)

1. for \( i \leftarrow 1, \ldots, 9 \) do  
2. \hspace{1em} for \( j \leftarrow 1, \ldots, 9 \) do  
3. \hspace{2em} if \( P_{i,j} \) is given then  
4. \hspace{3em} AddRow\((P_{i,j}, S)\)  
5. \hspace{2em} else  
6. \hspace{3em} for \( P_{i,j} \leftarrow 1, \ldots, 9 \) do  
7. \hspace{4em} AddRow\((P_{i,j}, S)\)  
8. \hspace{2em} end  
9. \hspace{1em} end  
10. end  
11. Delete column \( j \) from \( S \)

**Example 5.13** (Select poised lattice). If \( T^1_\Delta \) is given, we need a way to add points to get \( T^{n+1}_\Delta \).

The **dots** are \( T^1_\Delta \) is given and **crosses** are points we can add to get \( T^2_\Delta \).

There are four kinds of **shadings** which depends on which side we want to extend of the row and column that have most number of points.

In each kind of shading, we reduce it to the exact cover problem. These crosses can only be chosen once for every rows and columns.

From the figure above, we can get a matrix of exact cover problem.

\[
\begin{pmatrix}
1 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 \\
\end{pmatrix}
\]

The shading is \( 3 \times 3 \), the number of matrix’s columns is \( 3 + 3 \). The shading contains \( 4 \) crosses, the number of matrix’s rows is \( 4 \).