Appendix A

Derivation of the Proton Linkage Model

The derivation given here is taken from Laskowski & Finkenstadt, 1972 and was slightly modified without changing the basic principles. The original derivation was, however, done earlier (Wyman, 1964). I give it here for the sake of completeness.

Given is the following association reaction:

$$\mathrm{A} + \mathrm{B} \rightleftharpoons \mathrm{AB} \tag{A.1}$$

The equilibrium constant is

$$K_A = \frac{[\mathrm{AB}]}{[\mathrm{A}][\mathrm{B}]} \tag{A.2}$$

Each of the species A, B, and AB can bind protons. The total concentration of A is then given by

$$[\mathrm{A}] = \sum_{i=0}^{n_\mathrm{A}} [\mathrm{AH}_{n_\mathrm{A}-i}] \tag{A.3}$$

Analogous expressions are used for [B] and [AB], respectively. Then eq A.2 can be rewritten as

$$K_A = \frac{\sum_{i=0}^{n_\mathrm{AB}} [\mathrm{ABH}_{n_\mathrm{AB}-i}]}{\sum_{i=0}^{n_\mathrm{A}} [\mathrm{AH}_{n_\mathrm{A}-i}] \sum_{i=0}^{n_\mathrm{B}} [\mathrm{BH}_{n_\mathrm{B}-i}]} \tag{A.4}$$

where n_A, n_B, and n_AB are the numbers of titratable groups of A, B, and AB, respectively. The equilibrium between the fully-protonated species is defined as reference[1].

$$\mathrm{A}_{n_\mathrm{A}} + \mathrm{B}_{n_\mathrm{B}} \rightleftharpoons \mathrm{AB}_{n_\mathrm{AB}} \tag{A.5}$$

[1] Laskowski & Finkenstadt, 1972 allow that also additional proton binding sites can be created upon association. If however two protein associate, the number of proton binding sites does not change. While Laskowski & Finkenstadt, 1972 more generally assume that the number of proton binding sites in the docked complex is given by $n_\mathrm{AB} = c + n_\mathrm{A} + n_\mathrm{B}$, where c represents the number of proton binding sites created upon association, I assume here that $n_\mathrm{AB} = n_\mathrm{A} + n_\mathrm{B}$. The extension to the form used in Laskowski & Finkenstadt, 1972 is in principle possible, but makes the derivation more complicated.

The equilibrium constant for the reaction of the fully-protonated species is

$$K_A^o = \frac{[\mathrm{AB}_{n_\mathrm{AB}}]}{[\mathrm{A}_{n_\mathrm{A}}][\mathrm{B}_{n_\mathrm{B}}]} \tag{A.6}$$

Now an ionization constant as shown for A in eq A.7 is introduced for each species.

$$[\mathrm{AH}_{n_\mathrm{A}}] \rightleftharpoons [\mathrm{AH}_{n_\mathrm{A}-i}] + i[\mathrm{H}^+]; \quad \mathrm{L_{0A}} = 1; \quad \mathrm{L_{iA}} = \frac{[\mathrm{AH_{n_A-i}}][\mathrm{H^+}]^\mathrm{i}}{[\mathrm{AH_{n_A}}]} \tag{A.7}$$

Using eq A.6 and eq A.7, eq A.4 becomes

$$K_A = K_A^o \frac{\sum_{i=0}^{n_\mathrm{AB}} [\mathrm{H}^+]^{-i} L_{i\mathrm{AB}}}{\sum_{i=0}^{n_\mathrm{A}} [\mathrm{H}^+]^{-i} L_{i\mathrm{A}} \sum_{i=0}^{n_\mathrm{B}} [\mathrm{H}^+]^{-i} L_{i\mathrm{B}}} \tag{A.8}$$

Taking the logarithm of both sides of eq A.8 and differentiating with respect to $\mathrm{pH} = -\lg[H^+]$, we get

$$\frac{\mathrm{d}\lg K_A}{\mathrm{dpH}} = \frac{\mathrm{d}\lg(K_A^o)}{\mathrm{dpH}} + \frac{\mathrm{d}\lg(\sum_{i=0}^{n_\mathrm{AB}} [\mathrm{H}^+]^{-i} L_{i\mathrm{AB}})}{\mathrm{dpH}} - \frac{\mathrm{d}\lg(\sum_{i=0}^{n_\mathrm{A}} [\mathrm{H}^+]^{-i} L_{i\mathrm{A}})}{\mathrm{dpH}} - \frac{\mathrm{d}\lg(\sum_{i=0}^{n_\mathrm{B}} [\mathrm{H}^+]^{-i} L_{i\mathrm{B}})}{\mathrm{dpH}} \tag{A.9}$$

The derivation of the last three terms is demonstrated for A

$$\begin{aligned}
\frac{\mathrm{d}\lg(\sum_{i=0}^{n_\mathrm{A}} [\mathrm{H}^+]^{-i} L_{i\mathrm{A}})}{\mathrm{dpH}} &= \frac{-[H^+]}{\sum_{i=0}^{n_\mathrm{A}} [\mathrm{H}^+]^{-i} L_{i\mathrm{A}}} \frac{\mathrm{d}\sum_{i=0}^{n_\mathrm{A}} [\mathrm{H}^+]^{-i} L_{i\mathrm{A}}}{\mathrm{d}[\mathrm{H}^+]} \\
&= \frac{[H^+] \sum_{i=0}^{n_\mathrm{A}} i [\mathrm{H}^+]^{-i-1} L_{i\mathrm{A}}}{\sum_{i=0}^{n_\mathrm{A}} [\mathrm{H}^+]^{-i} L_{i\mathrm{A}}} \\
&= \langle i_\mathrm{A} \rangle
\end{aligned} \tag{A.10}$$

where $\langle i_\mathrm{A} \rangle$ represent respectively the average number of protons released from the fully-protonated form at the pH of interest. Therefore, we obtain

$$\frac{\mathrm{d}\lg K_A}{\mathrm{dpH}} = \overline{q} = \langle i_\mathrm{AB} \rangle - \langle i_\mathrm{A} \rangle - \langle i_\mathrm{B} \rangle \tag{A.11}$$

where $\langle i_\mathrm{A} \rangle$, $\langle i_\mathrm{B} \rangle$, and $\langle i_\mathrm{AB} \rangle$ represents the average number of protons released from the fully-protonated form of A, B, or AB at the pH of interest. Therefore, $\overline{q}$ is the number of protons released upon association. The reaction equilibrium at a defined pH is then given by

$$\mathrm{AH}_{n_\mathrm{A}-\langle i_\mathrm{A} \rangle} + \mathrm{BH}_{n_\mathrm{B}-\langle i_\mathrm{B} \rangle} \rightleftharpoons \mathrm{ABH}_{n_\mathrm{AB}-\langle i_\mathrm{AB} \rangle} + (\langle i_\mathrm{AB} \rangle - \langle i_\mathrm{A} \rangle - \langle i_\mathrm{B} \rangle)\mathrm{H}^+ \tag{A.12}$$

Integrating eq A.11 over pH results in

$$\lg K_A(\mathrm{pH}_2) - \lg K_A(\mathrm{pH}_1) = \int_{\mathrm{pH}_1}^{\mathrm{pH}_2} \overline{q}\mathrm{dpH} \tag{A.13}$$

Therefore, if the association constant at pH_1 is known, the association constant at pH_2 is supposed to be given by the expression above. An analogous expression for the pH dependence of redox potential can be easily derived.

Appendix B

Cluster Algorithm

Cluster analysis is used to relate objects on the basis of an object specific quantity using a defined distance measure. A large variety of cluster algorithms exists (Jain & Dubes, 1988). Some of them have been applied to cluster molecular conformations (Torda & van Gunsteren, 1994; Shenkin & McDonald, 1994; Boutonnet *et al.*, 1995). Here, I describe a new type of algorithm that Ernst-Walter Knapp and I developed and implemented during my PhD work to cluster molecular conformations on the basis of their mean square deviations (Ullmann *et al.*, 1998b).

B.1 Scoring Function

Given is a set of M molecular structures $\{\vec{x}_i\}$, $i = 1, \ldots, M$. Each structure $\vec{x}_i$ consists of n atoms $\vec{x}_i(j)$; $\vec{x}_i = (\vec{x}_i(1), \ldots, \vec{x}_i(n))$. This set can be subdivided into N clusters. Eq B.1 defines the average structure $\langle\vec{x}\rangle_k$ of cluster k with m_k elements.

$$\langle\vec{x}\rangle_k = \frac{1}{m_k}\sum_{i=1}^{m_k}\vec{x}_i; \qquad \sum_{k=1}^{N} m_k = M \tag{B.1}$$

The average structure of a cluster can also be interpreted as the center of this cluster. The squared distance $d^2(o,p)$ between two structures o and p is given by eq B.2.

$$d^2(o,p) = \frac{1}{n}\sum_{j=1}^{n}\left(\vec{x}_o(j) - \vec{x}_p(j)\right)^2 \tag{B.2}$$

where $\vec{x}_o(j)$ designates the j-th atom of structure o, n is the number of atoms in the structure. The variance of cluster k $\langle\Delta x^2\rangle_k$ is defined as the mean square distance between the elements of the cluster and the cluster center (eq B.3).

$$\begin{aligned}\langle\Delta x^2\rangle_k &= \frac{1}{m_k}\sum_{p=1}^{m_k}\frac{1}{n}\sum_{i=1}^{n}\left(\vec{x}_p(i) - \langle\vec{x}(i)\rangle_k\right)^2 \\ &= \frac{1}{m_k}\sum_{p=1}^{m_k} d^2(p,\langle k\rangle) \qquad \text{(B.3)} \\ & \qquad \text{(B.4)}\end{aligned}$$

where m_k is the number of elements of cluster k. The squared distance between two clusters k and l can be defined as the distance of the centers of these two clusters as $d^2(\langle k\rangle, \langle l\rangle)$ given in eq B.2.

Now, we define the scoring function S in eq B.5,

$$S = a \sum_{i=1}^{N} \langle \Delta x^2 \rangle_i + \frac{N-1}{M-1} \langle \Delta x^2 \rangle \tag{B.5}$$

where a is a tuning parameter to vary the average cluster size, n is the number of clusters, m is the total number of structures, $\langle \Delta x^2 \rangle_i$ is the variance of cluster i, and $\langle \Delta x^2 \rangle$ is the total variance over all structures. Minimization of this scoring function leads to a small variance within the clusters. The number of clusters becomes, however, not too large due to the second term in eq B.5.

An implementation of the equations outlined above would require an update of the average structure after each step. The variance must therefore be calculated from scratch after each step. The mean square deviation between the center of the cluster and each of its elements must be recalculated, because the center of the cluster changes. This would lead to an enormous computational burden. It can, however, be shown that

$$\begin{aligned} \langle \Delta x^2 \rangle_k &= \frac{1}{m_k} \sum_{p=1}^{m_k} d^2(p, \langle k \rangle) \\ &= \frac{1}{2{m_k}^2} \sum_{p=1}^{m_k} \sum_{o=1}^{m_k} d^2(p, o) \end{aligned} \tag{B.6}$$

The reformulation of the eq B.3 in eq B.6 allows the use of a precalculated mean square deviation matrix in the cluster algorithm. The variance of each cluster can be updated and must not be calculated from scratch after each step.

B.2 Data Organization and Implementation

The cluster algorithm is written in the programming language C (Kernighan & Ritchie, 1990). To implement the cluster algorithm efficiently, it is required to add and remove elements from a list. For that purpose, we developed a double-linked list, which makes it possible to remove elements even from the middle of a linked list and repair the remaining list. The clusters are grouped in a double-linked list (Figure B.1a). Each cluster is a structure that contains a pointer `prev` pointing to the previous cluster, a pointer `next` pointing to the next cluster, a pointer `Element` pointing to a double-linked list of elements, an integer `no_elements` that counts the number of elements of the cluster, and a real number `variance` that contains the variance of the cluster.

```
struct Cluster_struct {struct Element_struct    *element;
                        struct Cluster_struct    *prev;
                        struct Cluster_struct    *next;
                        int                      no_elements;
                        double                   variance;
                      };
```

The pointer `prev` of the first cluster and the pointer `next` of the last cluster point to `NULL`. If a cluster is removed from a double-linked list, the pointer are redirected.

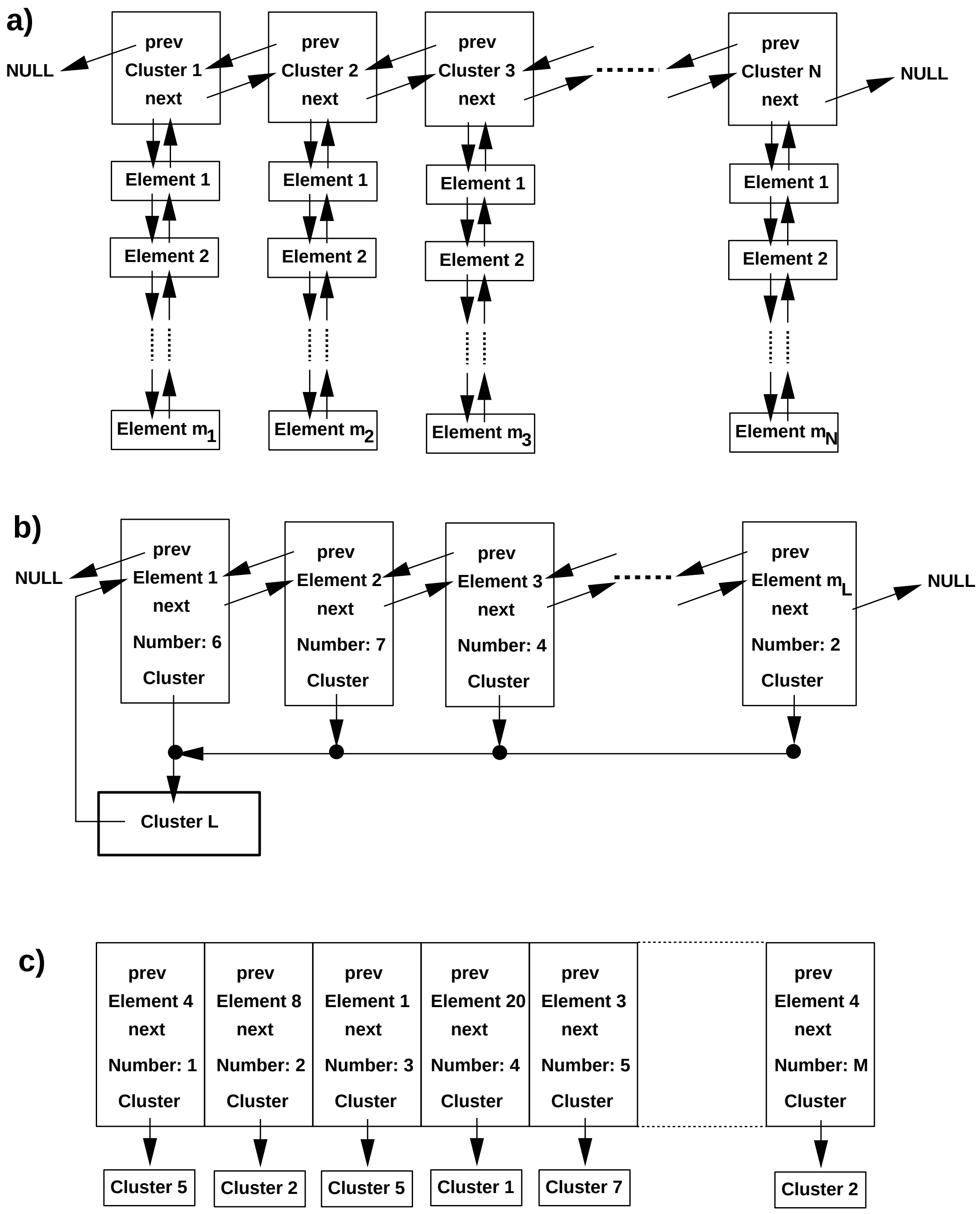

Figure B.1: Data organization for the cluster algorithm. a) Double-linked list of clusters. To each cluster a double-linked list of elements is assigned. b) Double-linked list of elements assigned to each cluster. Each element contains a pointer to that cluster to which it is assigned. c) One-dimensional array of elements. The number assigned to each element corresponds to the position of this element in the one-dimensional array.

```
if ((*cluster).next!=NULL) /* if not at the end of the list */
   {help=(*cluster).next;
    (*help).prev=(*cluster).prev;
   }
if ((*cluster).prev!=NULL) /* if not the top of the list */
   {help=(*cluster).prev;
    (*help).next=(*cluster).next;
    /* search for the top of the list */
    while ((*help).prev!=NULL) {help=(*help).prev;}
   }
else {help=(*cluster).next; /* help is now the top of the list */
     }
```

After this procedure, the variable `help` points to the top of the double-linked list of clusters. A new cluster is always added to the top of the double-linked list.

```
(*top).prev=new_cluster; /* Pointer redirection  */
(*new_cluster).next=top;
```

The double-linked list of elements is constructed similarly to the double-linked list of clusters, i. e., each element is a structure that contains two pointers, `prev` and `next` pointing to the previous and the next cluster in the double-linked list respectively. Besides, each element has a pointer `cluster` to the cluster to which the element is assigned and. an integer `number` that assigns this element to a position in an one dimensional array (Figure B.1c).

```
struct Element_struct {int                      number;
                       struct Cluster_struct    *cluster;
                       struct Element_struct    *prev;
                       struct Element_struct    *next;
                      };
```

Addition and elimination of an element to or from the double-linked list is done analogously to the addition and elimination of cluster to or from the double-linked list show above. This integer is required to access the mean square deviation in the precalculated mean square deviation matrix.

The algorithm uses several elementary steps: it moves an element from one cluster to another, it merges two clusters, it divides one cluster into two new clusters, and it sprinkles one cluster, i. e., assigns the elements of one cluster to other clusters. A step is accepted or rejected according to the Metropolis criterion (Metropolis *et al.*, 1953) in order to avoid to get trapped in a local minimum. Figure B.2 shows the flow chart of the program. The following commands can be currently combined in the program.

distribute_elements. This command generates the initial distribution of the elements to clusters. In the first step, one cluster is created and the first element is put into this cluster. The other elements are put into the clusters already created in the proceeding steps of this routine if the mean square deviation between this element and all elements in the cluster is smaller than a given threshold, otherwise it is put into a newly-created cluster.

random_element_move. This command moves a randomly-selected element to a randomly-selected cluster.

systematic_element_move1. This command moves a randomly-selected element to the cluster that has the smallest distance to the selected element but is not the cluster to which the element is assigned.

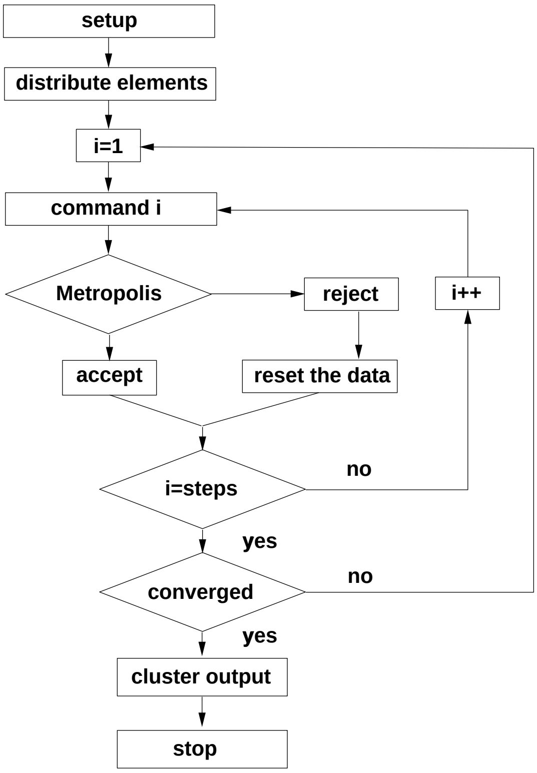

Figure B.2: Flow Chart of the Cluster Algorithm Program. A step is accepted or rejected according to the Metropolis criterion. The clustering is iterated until convergence is reached, i. e., until the difference between the old and the new score is smaller than a given threshold.

random_cluster_divide. This command divides a randomly-selected cluster. The elements that have the largest distance in the selected cluster are put into two newly-created separate clusters. The other elements are sorted to one of these two clusters according to their mean square deviation from the first element in the newly-created cluster.

systematic_cluster_divide. This command divides the cluster with the largest variance. The actual divide occurs in the same way as in random_cluster_divide.

random_cluster_sprinkle. This command sprinkles a randomly-selected cluster, i. e., its elements are assigned to the cluster to that the elements have the smallest distance. The selected cluster is eliminated after this move.

limit_cluster_sprinkle. This command sprinkles a randomly-selected cluster if the number of elements is smaller than a given number.

systematic_cluster_sprinkle. This command sprinkles the cluster with the largest variance.

random_cluster_merging. This command merges two randomly-selected clusters.

systematic_cluster_merging. This command merges the clusters which have the smallest distance from each other.

repeat_until_convergence. The last `N` step of the script are repeated until the difference between the old score and the new score is smaller than a given threshold, i. e., until the run converged.

Appendix C

Sequences

All sequences were obtained from SWISSPROT.

Cytochrome *f*

	60 70
Higher Plants	
Brassica rapa (turnip)	...VLANG**K**KGALN...
Oenothera hookeri (primose)	...VLANG**K**KGGLN...
Nicotiana tabacum (tobacco)	...VLANG**K**RGGLN...
Spinacia oleracea (spinach)	...VLANG**K**KGGLN...
Glycine max (soybean)	...VLANG**K**KGALN...
Pisum sativum (graden pea)	...VLANG**K**KGALN...
Vicia faba (broad bean)	...VLANG**K**KGALN...
Triticum aestivum (wheat)	...VLANG**K**KGGLN...
Zea mays (maize)	...VLANG**K**KGGLN...
Oryza sativa (rice)	...VLANG**K**KGGLN...
Pinus thunbergii (green pine)	...VLANG**K**KGALN...
Marchantia polymorpha	...VLANG**K**KGSLN...
Eukaryotic Algae	
Chlorophyta (Green Algae)	
Chlamydomonas reinhardtii	...VLANG**K**KGDLN...
Bacillariophyta (Diatoms)	
Odontella sinensis	...VGANG**K**KADLN...
Glaucophyta	
Cyanophora paradoxa	...IQANGQKGPLN...
Rhodophyta (Red Algae)	
Porphyra purpurea	...ILGNGSKGGLN...
Cyanobacteria	
Synechocystis (Strain PCC 6803)	...VLGDGSKGGLN...
Synechococcus (Strain PCC 7002)	...VLGDGSKGGLN...
Nostoc (Strain PCC 7906)	...VGADGSKVGLN...
Phormidium laminosum	...VQADGSKGPLN...

Residues 60 through 70 in all known sequences of cytochrome *f*. The residue Lys65 is missing in the cytochromes *f* from cyanobacteria and some primitive eukaryotic algae.

Cytochrome c_6

	50 60
Eukaryotic Algae	
Chlorophyta (Green Algae)	
Chlamydomonas reinhardtii	...I**Y**QVENGKGAM...
Monoraphidium braunii	...V**Y**QIENGKGAM...
Bryopsis maxima	...TSQVRNGKGAM...
Euglenophyta	
Euglena gracilis	...E**Y**QVRNGKGPM...
Euglena viridis	...E**Y**QVRNGKGPM...
Phaeophyta (Brown Algae)	
Petalonia fascia	...T**Y**QVTNGKNAM...
Alaria esculenta	...T**Y**QVTNGKNAM...
Rhodophyta (Red Algae)	
Porphyra tenera	...T**Y**QVQNGKNAM...
Porphyra purpurea	...T**Y**QVTNGKNAM...
Xanthophyta (Yellow Algae)	
Bumilleriopsis filiformis	...T**Y**QVTNGKNAM...
Chrysophyta (Golden Algae)	
Monochrysis lutheri	...V**Y**QVTNGKNAM...
Cyanobacteria	
Spirulina maxima	...A**Y**QVTNGKNAM...
Synechococcus lividus	...I**Y**QVQHGKNAM...
Synechococcus sp.	...M**Y**QVQNGKNAM...
Synechococcus (Strain PCC 7942)	...TTQVTNGKGAM...
Synechocystis (Strain PCC 6803)	...VAQITNGNGAM...
Microcysis aeruginosa	...VTQVTKGMGAM...
Anacystis nidulans	...TTQVTNGKGAM...
Aphanizomenon flos−aquae	...GAQVTNGKNAM...
Plectonema boryanum	...IAQVTHGKGAM...
Anabaena variabilis	...VAQVTNGKGAM...
Anabaena (Strain PCC 7120)	...IAQVTNGKNAM...
Anabaena (Strain PCC 7937)	...IAQVTNGKNAM...

Residues 50 through 60 in all known sequences of cytochrome c_6. The residue Tyr51, which is aligned with the dipole moment of the protein, is replaces by non-aromatic amino acids in cytochrome c_6 of many species.

Cytochrome c_6

	60 70
Eukaryotic Algae	
Chlorophyta (Green Algae)	
Chlamydomonas reinhardtii	...MPA**W**AD**R**LSEE...
Monoraphidium braunii	...MPA**W**DG**R**LDED...
Bryopsis maxima	...MPA**W**SD**R**LDDE...
Euglenophyta	
Euglena gracilis	...MPA**W**EGVLSED...
Euglena viridis	...MPA**W**EGVLDES...
Phaeophyta (Brown Algae)	
Petalonia fascia	...MPA**F**GG**R**LSET...
Alaria esculenta	...MPA**F**GS**R**LAET...
Rhodophyta (Red Algae)	
Porphyra tenera	...MPA**F**GG**R**LVDE...
Porphyra purpurea	...MPA**F**GG**R**LVDE...
Xanthophyta (Yellow Algae)	
Bumilleriopsis filiformis	...MPA**F**GG**R**LSDS...
Chrysophyta (Golden Algae)	
Monochrysis lutheri	...MPA**F**GG**R**LEDD...
Cyanobacteria	
Spirulina maxima	...MPG**F**NG**R**LSPK...
Synechococcus lividus	...MPA**F**AG**R**LTDE...
Synechococcus sp.	...MPA**F**GG**R**LSEA...
Synechococcus (Strain PCC 7942)	...MPA**F**GS**K**LSAD...
Synechocystis (Strain PCC 6803)	...MPG**F**KG**R**ISDS...
Microcysis aeruginosa	...MPA**F**GG**R**LSAE...
Anacystis nidulans	...MPA**F**GA**K**LSAD...
Aphanizomenon flos−aquae	...MPA**F**GI**R**LKAE...
Plectonema boryanum	...MPA**F**KG**R**LSDD...
Anabaena variabilis	...MPA**F**KG**R**LKPE...
Anabaena (Strain PCC 7120)	...MPA**F**KG**R**LKPE...
Anabaena (Strain PCC 7937)	...MPA**F**KG**R**LKPD...

Residues 60 through 70 in all known sequences of cytochrome c_6. Only tryptophane (W) or phenylalanine (F) is found at position 63. The aromatic group may interact with cationic the side chain of arginine (R) or lysine (K) at position 66 to form a cation-π complex. This cationic residue is missing in the two Euglenophyta.

Plastocyanin

	80 90
Higher Plants	
Populus nigra (poplar)	...YSF**Y**CSPHQGA...
Phaseolus vulgaris (French bean)	...YSF**Y**CSPHQGA...
Vicia faba (broad bean)	...YKF**Y**CSPHQGA...
Pisum sativum (graden pea)	...YKF**Y**CSPHQGA...
Nicotiana tabacum (tobacco)	...YTF**Y**CAPHQGA...
Solanum tuberosum (potato)	...YTF**Y**CAPHQGA...
Solanum crispum (potato tree)	...YSF**Y**CSPHQGA...
Lycopersicon esculentum (tomato)	...YTF**Y**CAPHQGA...
Lactuca sativa (lettuce)	...YSF**Y**CAPHQGA...
Capsella bursa-pastoris	...YSF**Y**CAPHQGA...
Arabidopsis thaliana (cress)	...YGF**Y**CAPHQGA...
Silene pratensis (white campion)	...YKF**Y**CAPHAGA...
Spinacia oleracea (spinach)	...YKF**Y**CSPHQGA...
Cucumis sativus (cucumber)	...YSF**Y**CSPHQGA...
Cucurbita pepo (squash)	...YSF**Y**CSPHQGA...
Petroselinum crispum (parsley)	...YKF**Y**CEPHAGA...
Daucus carota (carrot)	...YKF**Y**CEPHAGA...
Mercurialis perennis	...YSF**Y**CSPHQGA...
Sambucus nigra (european elder)	...YKF**Y**CSPHQGA...
Rumex obtusifolius (bitter dock)	...YSF**Y**CSPHQGA...
Hordeum vulgare (barley)	...YGF**Y**CEPHAGA...
Oryza sativa (rice)	...YGF**Y**CEPHAGA...
Eukaryotic Algae	
Chlorophyta (Green Algae)	
Chlamydomonas reinhardtii	...YGY**Y**CEPHQGA...
Chlorella fusca	...YGY**F**CEPHQGA...
Scenedesmus obliquus	...YGY**F**CEPHQGA...
Ulva arasakii	...YGV**Y**CEPHAGA...
Enteromorpha prolifera	...YGV**Y**CDPHSGA...
Cyanobacteria	
Prochlorothrix hollandica	...YSF**Y**CTPH**R**GA...
Synechocystis (Strain PCC 6803)	...YTY**Y**CEPH**R**GA...
Anabaena variabilis	...YTF**Y**CEPH**R**GA...
Anabaena (Strain PCC 7937)	...YSF**Y**CEPH**R**GA...
Anabaena (Strain PCC 7120)	...YTF**Y**CEPH**R**GA...
Phormidium laminosum	...YTY**Y**CAPH**R**GA...

Residues 80 through 90 in all known sequences of plastocyanin. The tyrosine (Y) at position 83 is replaced by a phenylalanine (F) in two of the sequences. The aromatic group may interact with the cationic side chain of Arg88, which is conserved in cyanobacterial plastocyanin, to form a cation-π complex.