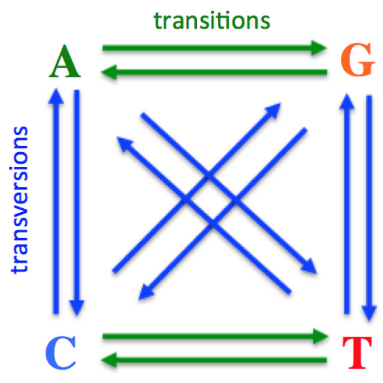
Exploring the Genomic Cross-Mutation Network

I. The General Case of Monomers in a Single-Stranded Genome

Let us consider the inter-conversion network of the DNA bases, where we consider the overall genomic fractions of the individual base types (A, T, G, C).



We can start by first examining the differential equation formulation of the cross-mutation network for individual bases in its most general form, where the mutation rate constants are all unique (non-symmetric case). This would correspond to the folowing scheme:

		To			
		A	G	T	C
From	A	–	$\mathbf{k}_{AG}$	$\mathbf{k}_{AT}$	$\mathbf{k}_{AC}$
	G	$\mathbf{k}_{GA}$	–	$\mathbf{k}_{GT}$	$\mathbf{k}_{GC}$
	T	$\mathbf{k}_{TA}$	$\mathbf{k}_{TG}$	–	$\mathbf{k}_{TC}$
	C	$\mathbf{k}_{CA}$	$\mathbf{k}_{CG}$	$\mathbf{k}_{CT}$	–

mutation rate constants

$$\begin{cases} dC_A/dt = \mathbf{k}_{CA}C_C+\mathbf{k}_{TA}C_T+\mathbf{k}_{GA}C_G-(\mathbf{k}_{AC}+\mathbf{k}_{AT}+\mathbf{k}_{AG})\,C_A \\ dC_G/dt = \mathbf{k}_{AG}C_A+\mathbf{k}_{CG}C_C+\mathbf{k}_{TG}C_T-(\mathbf{k}_{GA}+\mathbf{k}_{GT}+\mathbf{k}_{GC})\,C_G \\ dC_T/dt = \mathbf{k}_{AT}C_A+\mathbf{k}_{GT}C_G+\mathbf{k}_{CT}C_C-(\mathbf{k}_{TA}+\mathbf{k}_{TC}+\mathbf{k}_{TG})\,C_T \\ dC_C/dt = \mathbf{k}_{AC}C_A+\mathbf{k}_{TC}C_T+\mathbf{k}_{GC}C_G-(\mathbf{k}_{CA}+\mathbf{k}_{CT}+\mathbf{k}_{CG})\,C_C \end{cases}$$

The model is defined via 4 coupled equations, each describing the evolution of A, G, T and C base contents (in fractions). In the *Mathematica* program, the above system can be written as:

```
(*   DSolve[{CA'[t]==kCA*CC[t]+kTA*CT[t]+kGA*CG[t]-(kAC+kAT+kAG)*CA[t],
             CG'[t]==kAG*CA[t]+kCG*CC[t]+kTG*CT[t]-(kGA+kGT+kGC)*CG[t],
             CT'[t]==kAT*CA[t]+kGT*CG[t]+kCT*CC[t]-(kTA+kTC+kTG)*CT[t],
             CC'[t]==kAC*CA[t]+kTC*CT[t]+kGC*CG[t]-(kCA+kCT+kCG)*CC[t],
             CA[0]==CA0,
             CG[0]==CG0,
             CT[0]==CT0,
             CC[0]==CC0},{CA,CG,CT,CC},t]//FullSimplify    *)
```

The solution is too complex and intractable. We can study the behaviour of the system by exploring its numerical solutions with variable rate constants (see the R program <ATGC Dynamics Solver>). In particular, one can see that the time required for the equillibration of the system depends on k_{ij} rate constants and the C_{i0} initial base contents.

We can, however, obtain the symbolic solution of the system at equillibrium. There, the base contents are supposed to stay constant, hence we need to solve the system of equations displayed below, additionally setting the sum of all the base contents (in fractions) to 1.

```
Solve[kCA * CC + kTA * CT + kGA * CG - (kAC + kAT + kAG) * CA == 0 &&
      kAG * CA + kCG * CC + kTG * CT - (kGA + kGT + kGC) * CG == 0 &&
      kAT * CA + kGT * CG + kCT * CC - (kTA + kTC + kTG) * CT == 0 &&
      kAC * CA + kTC * CT + kGC * CG - (kCA + kCT + kCG) * CC == 0 &&
      CA + CT + CG + CC == 1,
      {CA, CG, CT, CC}] // FullSimplify
```

```
{{CA → (kCT (kGA + kGC + kGT) kTA + kCA (kGA + kGC + kGT) (kTA + kTC) +
        kCT kGA kTG + kCA (kGA + kGC) kTG + kCG (kGT kTA + kGA (kTA + kTC + kTG))) /
      (kAG kCT kGC + kAG kCA kGT + kAG kCG kGT + kAG kCT kGT + kAG kCA kTA +
        kAG kCG kTA + kAG kCT kTA + kCA kGA kTA + kCG kGA kTA + kCT kGA kTA + kAG kGC kTA +
        kCA kGC kTA + kCT kGC kTA + kCA kGT kTA + kCG kGT kTA + kCT kGT kTA + kAG kCA kTC +
        kAG kCG kTC + kCA kGA kTC + kCG kGA kTC + kAG kGC kTC + kCA kGC kTC + kAG kGT kTC +
        kCA kGT kTC + ((kCA + kCG + kCT) (kAG + kGA) + (kAG + kCA) kGC) kTG +
        kAT ((kGA + kGC + kGT) (kCT + kTC) + (kCT + kGC) kTG + kCA (kGA + kGC + kGT + kTG) +
            kCG (kGA + kGT + kTC + kTG)) + kAC ((kGA + kGC + kGT) (kCT + kTA + kTC) +
            (kCT + kGA + kGC) kTG + kCG (kGT + kTA + kTC + kTG))),
  CG → (kAT kCG kTC + kAC kCG (kTA + kTC) + kAC (kCG + kCT) kTG + kAT (kCA + kCG + kCT) kTG +
        kAG (kCT (kTA + kTG) + kCA (kTA + kTC + kTG) + kCG (kTA + kTC + kTG))) /
      (kAG kCT kGC + kAG kCA kGT + kAG kCG kGT + kAG kCT kGT + kAG kCA kTA +
        kAG kCG kTA + kAG kCT kTA + kCA kGA kTA + kCG kGA kTA + kCT kGA kTA + kAG kGC kTA +
        kCA kGC kTA + kCT kGC kTA + kCA kGT kTA + kCG kGT kTA + kCT kGT kTA + kAG kCA kTC +
        kAG kCG kTC + kCA kGA kTC + kCG kGA kTC + kAG kGC kTC + kCA kGC kTC + kAG kGT kTC +
        kCA kGT kTC + ((kCA + kCG + kCT) (kAG + kGA) + (kAG + kCA) kGC) kTG +
        kAT ((kGA + kGC + kGT) (kCT + kTC) + (kCT + kGC) kTG + kCA (kGA + kGC + kGT + kTG) +
            kCG (kGA + kGT + kTC + kTG)) + kAC ((kGA + kGC + kGT) (kCT + kTA + kTC) +
            (kCT + kGA + kGC) kTG + kCG (kGT + kTA + kTC + kTG))),
  CT → (kAG kCT kGC + kAC kCT (kGA + kGC) + kAC (kCG + kCT) kGT + kAG (kCA + kCG + kCT) kGT +
        kAT (kCG (kGA + kGT) + kCA (kGA + kGC + kGT) + kCT (kGA + kGC + kGT))) /
      (kAG kCT kGC + kAG kCA kGT + kAG kCG kGT + kAG kCT kGT + kAG kCA kTA +
        kAG kCG kTA + kAG kCT kTA + kCA kGA kTA + kCG kGA kTA + kCT kGA kTA + kAG kGC kTA +
        kCA kGC kTA + kCT kGC kTA + kCA kGT kTA + kCG kGT kTA + kCT kGT kTA + kAG kCA kTC +
        kAG kCG kTC + kCA kGA kTC + kCG kGA kTC + kAG kGC kTC + kCA kGC kTC + kAG kGT kTC +
        kCA kGT kTC + ((kCA + kCG + kCT) (kAG + kGA) + (kAG + kCA) kGC) kTG +
        kAT ((kGA + kGC + kGT) (kCT + kTC) + (kCT + kGC) kTG + kCA (kGA + kGC + kGT + kTG) +
            kCG (kGA + kGT + kTC + kTG)) + kAC ((kGA + kGC + kGT) (kCT + kTA + kTC) +
            (kCT + kGA + kGC) kTG + kCG (kGT + kTA + kTC + kTG))),
  CC → (kAT (kGA + kGC + kGT) kTC + kAC (kGA + kGC + kGT) (kTA + kTC) + kAT kGC kTG +
        kAC (kGA + kGC) kTG + kAG (kGT kTC + kGC (kTA + kTC + kTG))) /
      (kAG kCT kGC + kAG kCA kGT + kAG kCG kGT + kAG kCT kGT + kAG kCA kTA +
        kAG kCG kTA + kAG kCT kTA + kCA kGA kTA + kCG kGA kTA + kCT kGA kTA + kAG kGC kTA +
        kCA kGC kTA + kCT kGC kTA + kCA kGT kTA + kCG kGT kTA + kCT kGT kTA + kAG kCA kTC +
        kAG kCG kTC + kCA kGA kTC + kCG kGA kTC + kAG kGC kTC + kCA kGC kTC + kAG kGT kTC +
        kCA kGT kTC + ((kCA + kCG + kCT) (kAG + kGA) + (kAG + kCA) kGC) kTG +
        kAT ((kGA + kGC + kGT) (kCT + kTC) + (kCT + kGC) kTG + kCA (kGA + kGC + kGT + kTG) +
            kCG (kGA + kGT + kTC + kTG)) + kAC ((kGA + kGC + kGT) (kCT + kTA + kTC) +
            (kCT + kGA + kGC) kTG + kCG (kGT + kTA + kTC + kTG)))}}
```

All the solutions, as function of the rate constants, are found. The lines below are for holding the equilibrium A, G, T and C contents in the C_{Aeq}, C_{Geq}, C_{Teq}, C_{Ceq} objects.

```
{CAeq, CGeq, CTeq, CCeq} = {CA, CG, CT, CC} /.
      Solve[kCA * CC + kTA * CT + kGA * CG - (kAC + kAT + kAG) * CA == 0 &&
            kAG * CA + kCG * CC + kTG * CT - (kGA + kGT + kGC) * CG == 0  &&
            kAT * CA + kGT * CG + kCT * CC - (kTA + kTC + kTG) * CT == 0  &&
            kAC * CA + kTC * CT + kGC * CG - (kCA + kCT + kCG) * CC == 0  &&
            CA + CT + CG + CC  ==  1,
           {CA, CG, CT, CC}] // FullSimplify // First;
```

Since most of the present day genomes comply the Chargaff's 2nd parity rule, we can postulate two conditions. First is the condition that the Chargaff's rule should be met at the equillibrium state of the system. Secondly, most of the genomes should have become equillibrated in terms of the Chargaff's second parity rule (note, that the "Chargaff equilibration" can occur sooner than the base content equilibration, see the <ATGC Dynamics Solver>) within the span of the age of Life (~4 billion years). For the time being, let us neglect the equilibration time constraint, which is difficult to account for analytically (we have explored it numerically in the paper), and concentrate on finding only the equillibrium solutions to the above problem. We can try to find dependencies in the rate constants, by imposing the Chargaff's second parity rule.

```
Solve[CAeq == CTeq &&
      CGeq ==  CCeq &&
      CAeq + CTeq + CGeq + CCeq == 1,
      {kTC, kCT, kCG, kGC, kAG, kGA, kAT, kTA, kCA, kAC, kGT, kTG}] //
 FullSimplify
```

```
Solve::svars : Equations may not give solutions for all "solve" variables. >>
```

$$\left\{\left\{kTC \to \frac{1}{kCA + kGA}(kAC\,(kCG + kCT - kGA - kGC) + (kCA + kCG + kCT - kGC)\,(kAG + kAT - kTA)),\right.\right.$$
$$\left. kTG \to \frac{1}{kCA + kGA}(kAG\,(-kCA - kCG + kGC + kGT) - (kCG - kGA - kGC - kGT)\,(kAC + kAT - kTA))\right\},$$

```
 {kGC → kCA + kCG + kCT, kGA → -kCA, kGT → -kCT}, {kGA → -kCA, kTA → kAC + kAG + kAT,
  kTG → (-kAG (kCA + kCG + kCT - kGC) + kAC (-kCA - kCG + kGC + kGT) +
       (-kCA - kCG + kGC + kGT) kTC) / (kCA + kCG + kCT - kGC)},
 {kGC → kCA + kCG + kCT, kGA → -kCA, kTA → kAC + kAG + kAT, kGT → -kCT},
 {kTC → -kAC, kGC → kCA + kCG + kCT, kGA → -kCA, kTA → kAC + kAG + kAT}}
```

This step returns four possible set of solutions for only a part of the requested variables. However, only the first set is applicable to our physical problem, as only that set complies with the fact that the rate constants should always be greater than 0, hence there cannot be two non-0 positive numbers that are negative to each other (such as in $k_{GA} = -k_{CA}$ etc. in some solution sets). Let us now assign the found interdependences into the special objects k_{TCch} and k_{TGch}.

```
{kTCch, kTGch} = {kTC, kTG} /.
      Solve[CAeq == CTeq &&
            CGeq ==  CCeq &&
            CAeq + CTeq + CGeq + CCeq == 1,
           {kTC, kCT, kCG, kGC, kAG, kGA, kAT, kTA, kCA, kAC, kGT, kTG}] //
    FullSimplify // First;
```

Now we can try to solve the equations for the rate constants, plugging in whatever we have already learned above. Unfortunately, trying multiple variations of the system, with different assumptions that conform our physical problem, does not converge into a new solution. Examples of the systems that have been tried to be solved are below:

```
(*   Solve[CAeq==CTeq &&
           CGeq== CCeq &&
           kTC==kTCch &&
           kTG ==kTGch &&
          {kTC, kCT, kCG, kGC, kAG, kGA, kAT, kTA, kCA, kAC, kGT, kTG}>0 &&
          (1/2)>{CAeq,CTeq, CGeq, CCeq}>0 &&
           CAeq+CTeq+CGeq+CCeq==1,
          {kTC, kCT, kCG, kGC, kAG, kGA, kAT, kTA, kCA, kAC, kGT, kTG}]//
 FullSimplify   *)
```

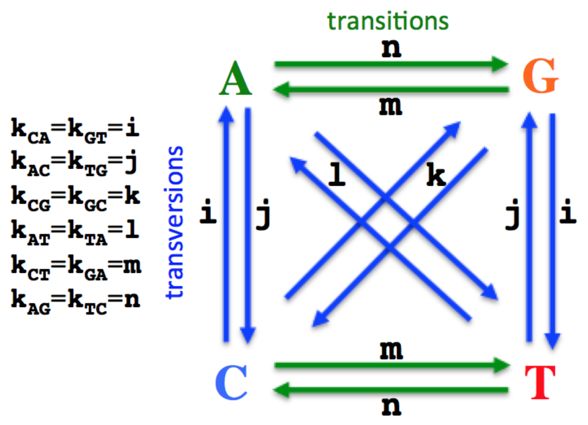
```
(*   Solve[kCA*GC/2+kTA*((1-GC)/2)+kGA*GC/2-(kAC+kAT+kAG)*((1-GC)/2)==0 &&

   kAG*((1-GC)/2)+kCG*GC/2+kTG*((1-GC)/2)-(kGA+kGT+kGC)*GC/2==0 &&
           kAT*((1-GC)/2)+kGT*GC/2+kCT*GC/2-(kTA+kTC+kTG)*((1-GC)/2)==0 &&
          kAC*((1-GC)/2)+kTC*((1-GC)/2)+kGC*GC/2-(kCA+kCT+kCG)*GC/2==0 &&
           {kTC, kCT, kCG, kGC, kAG, kGA, kAT, kTA, kCA, kAC, kGT, kTG} > 0 &&
           1 > GC > 0,
          {kTC, kCT, kCG, kGC, kAG, kGA, kAT, kTA, kCA, kAC, kGT, kTG}]//
 FullSimplify   *)
```

```
(*   Solve[CAeq==CTeq &&
           CGeq== CCeq &&
           {kTC, kCT, kCG, kGC, kAG, kGA, kAT, kTA, kCA, kAC, kGT, kTG}>0 &&
           {CAeq,CTeq, CGeq, CCeq}>0 &&
           CAeq+CTeq+CGeq+CCeq==1,
           {kTC, kCT, kCG, kGC, kAG, kGA, kAT, kTA, kCA, kAC, kGT, kTG}]//
 FullSimplify   *)
```

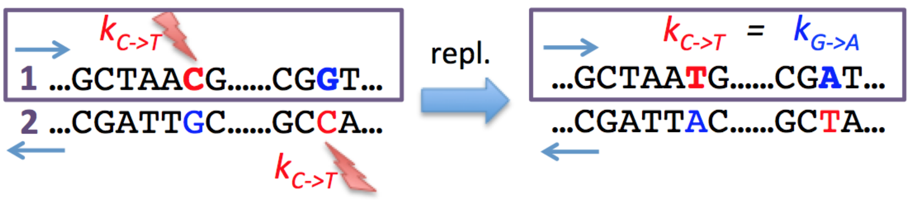
To sum up what has been done above; we have constructed the system of differential equations that describes the cross-mutation network among the four DNA bases. The model was defined in its most general form, where all the rate constants are unique. This system is applicable, for instance, to a single-stranded nucleic acids. The equations describe the time evolution of the overall genomiccontents of A, T, G and C bases. We solved the system at its equilibrium and found the exact solutions (rather complex ones) for the C_{ieq} equilibrium fractions of all i bases. However, one of the fundamental genomic relations, the Chargaff's second parity rule, is not following out of the found solutions. To continue the exploration of this system, we have next incorporated the second parity rule as an additional constraint in the equation. This enabled us to find an additional dependencies (k_{TC} and k_{TG} as a function of the other rate constants), that enables the system to comply the rule. We shall later see that the solutions we obtain by exploiting the regularities arising from the double strandedness of DNA are satisying all the general relations found in this section. Furthermore, in all the subsequently discussed systems, Chargaff's second parity rule directly follows from the solutions without its prior usage as a constraint.

2. The Case of Monomers in a Double-Stranded Genome

There is an important set of equalities in the rate constants that emerges because of the double-strandedness of DNA and the complementarity involved in the core of DNA replication. Even when considering the bases of only one strand, the following is true in a double-stranded DNA:



To clarify how these equalities emerge, consider the example for the $k_{CT} = k_{GA}$ in the figure below. Here and throughout the paper, by k_{CT} (and analogously by any k_{ij}) we mean the rate constant of the C:G to T:A conversion, which involves all the processes that fixate the mutation in both strands. With a very reasonable assumption of the C to T mutation rate being equal in both strands, whichever rate C converts to T in the strand of interest (enclosed in a red box), with the same rate C will convert to T in the opposite strand in other sites of the genome. Therefore, the strand of interest will have G (complementar to C) to A (complementar to T) conversions with the rate similar to C to T conversion.


$k_{C\text{->}T}$
1 ...GCTAACG......CGGT...
2 ...CGATTGC......GCCA...
$k_{C\text{->}T}$
repl.
$k_{C\text{->}T}$ = $k_{G\text{->}A}$
...GCTAATG......CGAT...
...CGATTAC......GCTA...

This symmetry in rate constants significantly simplifies the system of differential equations to describe the base content dynamics in a double-stranded DNA.

From \ To	A	G	T	C
A	–	n	l	j
G	m	–	i	k
T	l	j	–	n
C	i	k	m	–

mutation rate constants

$$\begin{cases} dC_A/dt = iC_C + lC_T + mC_G - (j+l+n)C_A \\ dC_G/dt = nC_A + kC_C + jC_T - (m+i+k)C_G \\ dC_T/dt = lC_A + iC_G + mC_C - (l+n+j)C_T \\ dC_C/dt = jC_A + nC_T + kC_G - (i+m+k)C_C \end{cases}$$

In *Mathematica*, the system can be specified as:

```
(*   DSolve[{ CA'[t]==i*CC[t]+l*CT[t]+m*CG[t]-(j+l+n)*CA[t],
             CG'[t]==n*CA[t]+k*CC[t]+j*CT[t]-(m+i+k)*CG[t],
             CT'[t]==l*CA[t]+i*CG[t]+m*CC[t]-(l+n+j)*CT[t],
             CC'[t]==j*CA[t]+n*CT[t]+k*CG[t]-(i+m+k)*CC[t],
             CA[0]== CA0,
             CG[0]== CG0,
             CT[0]== CT0,
             CC[0]== CC0},{CA,CG,CT,CC},t]//FullSimplify   *)
```

for which we can find the symbolic solution at equilibrium (see the <ATGC Dynamics Solver> for exploring the numerical solutions).

```
Solve[ i * CC + l * CT + m * CG - (j + l + n) * CA == 0 &&
      n * CA + k * CC + j * CT - (m + i + k) * CG == 0 &&
      l * CA + i * CG + m * CC - (l + n + j) * CT == 0 &&
      j * CA + n * CT + k * CG - (i + m + k) * CC == 0 &&
      CA + CT + CG + CC == 1,
     {CA, CG, CT, CC}] // FullSimplify
```

$$\left\{\left\{CA \to \frac{i+m}{2\ (i+j+m+n)},\ CG \to \frac{j+n}{2\ (i+j+m+n)},\ CT \to \frac{i+m}{2\ (i+j+m+n)},\ CC \to \frac{j+n}{2\ (i+j+m+n)}\right\}\right\}$$

As you can see, the system is fully solved, and the solutions imply the $C_A = C_T$ and $C_G = C_C$ equality. Therefore, Chargaff's second parity rule emerges as a result of the above discussed symmetry in rate constants that are a result of the double-stranded nature of DNA with complementary pairings.

Since later on we shall switch to more complex cases, still looking for symbolic solutions, below you can see some other ways of solving such systems of equations. All the below methods have different performance in *Mathematica,* hence will be tried for more complex systems for the search of the one that converges. In the examples below, you can make sure that all those approaches result into the same solution for the managable systems.

If we represent the equation system in a Ax=b matrix form, the solutions can be found using linear algerbra. First, we need to construct the matrices A and b from the equations. In *Mathematica*, we can use the memory efficient sparse array convention, since most of the entries in the matrices are expected to be 0s.

```
equations2 = { i * CC + l * CT + m * CG - (j + l + n) * CA == 0,
               n * CA + k * CC + j * CT - (m + i + k) * CG == 0,
               l * CA + i * CG + m * CC - (l + n + j) * CT == 0,
               j * CA + n * CT + k * CG - (i + m + k) * CC == 0,
               CA + CT + CG + CC == 1};

variables2 = {CA, CG, CT, CC};

matrix2 = CoefficientArrays[equations2, variables2];
A = matrix2[[2]];
b = -matrix2[[1]];
```

Since here the matrices are simpler, we can parse and view those in full, to make sure that everything is correct.

```
{MatrixForm[A], MatrixForm[b] }
```

$$\left\{\begin{pmatrix} -j-l-n & m & l & i \\ n & -i-k-m & j & k \\ l & i & -j-l-n & m \\ j & k & n & -i-k-m \\ 1 & 1 & 1 & 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}\right\}$$

Now we can solve the linear system using symbolic LinearSolve for $\{C_A, C_G, C_T, C_C\}$ variables:

```
LinearSolve[A, b] // FullSimplify
```

$$\left\{\frac{i+m}{2\,(i+j+m+n)}, \frac{j+n}{2\,(i+j+m+n)}, \frac{i+m}{2\,(i+j+m+n)}, \frac{j+n}{2\,(i+j+m+n)}\right\}$$

The LinearSolve function can be applied with many algorithmic Method arguments, each having different memory efficiency.

Another method of solving the system is by generating an augmented matrix equation [A|b][x] and doing a Gaussian elimination procedure to simplify it to the state with most 0s.

```
Aug = Transpose[Join[Transpose[A], {b}]];
```

```
result2 = RowReduce[Aug];
```

```
MatrixForm[result2]
```

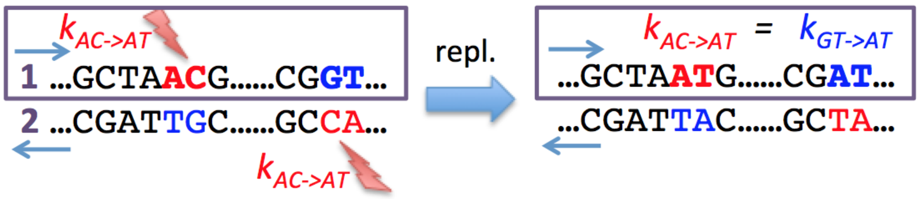
$$\begin{pmatrix} 1 & 0 & 0 & 0 & \frac{i+m}{2\,(i+j+m+n)} \\ 0 & 1 & 0 & 0 & \frac{j+n}{2\,(i+j+m+n)} \\ 0 & 0 & 1 & 0 & \frac{i+m}{2\,(i+j+m+n)} \\ 0 & 0 & 0 & 1 & \frac{j+n}{2\,(i+j+m+n)} \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

As you can see, the last column of the reduced matrix contains the solutions for $\{C_A, C_G, C_T, C_C\}$ variables.

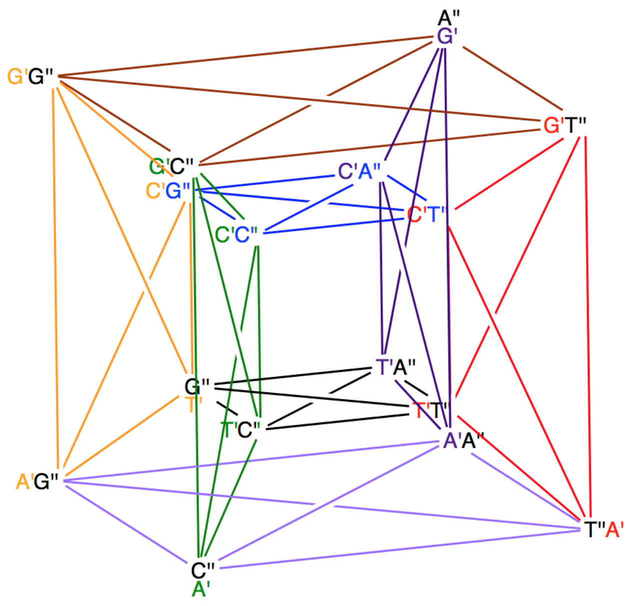
3. The Case of Dimers in a Double-Stranded Genome

It is straight-forward to infer that the same DNA-double-strand-driven complementarities in mutations produce similar equalities in rate constants regardless of the size of oligomer that we consider. In Section 2, the rate constants were reflecting the average base substitution accross different sequence context found in a genome. Therefore, the above model is applicable for finding the individual base contents in the scale of the entire genome. In contrats to this, when we want to recover the k-mer (k=1, 2, 3, 4, ...) frequencies in a given genome, we need to consider the rate constants for many more k-mer transitions, as the neighbouring bases alter the mutation rate at a given site. To clarify this, consider the CpA → TpA and CpG → TpG substitutions in the dimeric context. Both substitutions are a result of C → T point mutation. However the rates for the two substitutions will be different ($k_{CA2TA} \neq k_{CG2TG}$), because of the neighbouring base (A vs. G) effects. In this particular example, we know that in the CpG context, the mutation rates for C are substantially elevated.

In any case, the equalities between the rate constants noted in the previous section are true in the k-mer case as well, with the only difference being that we need to account directionalities (5’-3’ vs. 3’-5’) of the oligonucleotides. Hence, the rate constants of the substitutions that are reverse complementar to each other will be the same. You can see this on the example of ApC → ApT substitution in the figure below.


$k_{AC\text{->}AT}$
1 ...GCTAACG......CGGT...
2 ...CGATTGC......GCCA...
$k_{AC\text{->}AT}$
repl.
$k_{AC\text{->}AT}$ = $k_{GT\text{->}AT}$
...GCTAATG......CGAT...
...CGATTAC......GCTA...

Taking into account that even the oligomeric substitutions are pevalently driven by point mutations, we can construct the whole network of all possible dimer substitutions. The network is easy to visualise on the basis of the hypercube (tesseract, 3D projection of a 4D cube) geometric object. In the figure below, the superscript signs ‘ and “ mark the first and the second bases along the 5’-3’ direction. Each line in the scheme should be interpreted as a set of two arrows in opposite directions.



For this network, we can now construct the system of state equations by accounting for all the rate constant equalities, as described above for the dimeric case.

```
(*   DSolve[
      {CAA'[t] == kCA2AA*CCA[t]-kAA2CA*CAA[t] + kTA2AA*CTA[t]- kAA2TA*CAA[t]+ kGA2AA*CGA[t]- kAA2GA*CAA[t]+
     kAC2AA*CAC[t] - kAA2AC*CAA[t]+ kAT2AA*CAT[t]- kAA2AT*CAA[t]+ kAG2AA*CAG[t]- kAA2AG*CAA[t],
       CAC'[t] == kCC2AC*CCC[t]-kAC2CC*CAC[t] + kGA2GT*CTC[t]- kAC2TC*CAC[t]+ kGC2AC*CGC[t]- kAC2GC*CAC[t]+
     kAA2AC*CAA[t] - kAC2AA*CAC[t]+ kAT2AC*CAT[t]- kAC2AT*CAC[t]+ kAG2AC*CAG[t]- kAC2AG*CAC[t],
       CAG'[t] == kCG2AG*CCG[t]-kAG2CG*CAG[t] + kCA2CT*CTG[t]- kAG2TG*CAG[t]+ kCC2CT*CGG[t]- kAG2GG*CAG[t]+
     kAA2AG*CAA[t] - kAG2AA*CAG[t]+ kAC2AG*CAC[t]- kAG2AC*CAG[t]+ kAT2AG*CAT[t]- kAG2AT*CAG[t],
       CAT'[t] == kAG2AT*CCT[t]-kAT2AG*CAT[t] + kAA2AT*CTT[t]- kAT2AA*CAT[t]+ kAC2AT*CGT[t]- kAT2AC*CAT[t]+
     kAA2AT*CAA[t] - kAT2AA*CAT[t]+ kAC2AT*CAC[t]- kAT2AC*CAT[t]+ kAG2AT*CAG[t]- kAT2AG*CAT[t],
       CCA'[t] == kAA2CA*CAA[t]-kCA2AA*CCA[t] + kTA2CA*CTA[t]- kCA2TA*CCA[t]+ kGA2CA*CGA[t]- kCA2GA*CCA[t]+
     kCC2CA*CCC[t] - kCA2CC*CCA[t]+ kAG2TG*CCT[t]- kCA2CT*CCA[t]+ kCG2CA*CCG[t]- kCA2CG*CCA[t],
       CCC'[t] == kAC2CC*CAC[t]-kCC2AC*CCC[t] + kGA2GG*CTC[t]- kCC2TC*CCC[t]+ kGC2CC*CGC[t]- kCC2GC*CCC[t]+
     kCA2CC*CCA[t] - kCC2CA*CCC[t]+ kAG2GG*CCT[t]- kCC2CT*CCC[t]+ kCG2CC*CCG[t]- kCC2CG*CCC[t],
       CCG'[t] == kAG2CG*CAG[t]-kCG2AG*CCG[t] + kCA2CG*CTG[t]- kCG2CA*CCG[t]+ kCC2CG*CGG[t]- kCG2CC*CCG[t]+
     kCA2CG*CCA[t] - kCG2CA*CCG[t]+ kCC2CG*CCC[t]- kCG2CC*CCG[t]+ kAG2CG*CCT[t]- kCG2AG*CCG[t],
       CCT'[t] == kAT2AG*CAT[t]-kAG2AT*CCT[t] + kAA2AG*CTT[t]- kAG2AA*CCT[t]+ kAC2AG*CGT[t]- kAG2AC*CCT[t]+
     kCA2CT*CCA[t] - kAG2TG*CCT[t]+ kCC2CT*CCC[t]- kAG2GG*CCT[t]+ kCG2AG*CCG[t]- kAG2CG*CCT[t],
       CGA'[t] == kAA2GA*CAA[t]-kGA2AA*CGA[t] + kCA2GA*CCA[t]- kGA2CA*CGA[t]+ kTA2GA*CTA[t]- kGA2TA*CGA[t]+
     kGC2GA*CGC[t] - kGA2GC*CGA[t]+ kAC2TC*CGT[t]- kGA2GT*CGA[t]+ kCC2TC*CGG[t]- kGA2GG*CGA[t],
       CGC'[t] == kAC2GC*CAC[t]-kGC2AC*CGC[t] + kCC2GC*CCC[t]- kGC2CC*CGC[t]+ kGA2GC*CTC[t]- kGC2GA*CGC[t]+
     kGA2GC*CGA[t] - kGC2GA*CGC[t]+ kAC2GC*CGT[t]- kGC2AC*CGC[t]+ kCC2GC*CGG[t]- kGC2CC*CGC[t],
       CGG'[t] == kAG2GG*CAG[t]-kCC2CT*CGG[t] + kCG2CC*CCG[t]- kCC2CG*CGG[t]+ kCA2CC*CTG[t]- kCC2CA*CGG[t]+
     kGA2GG*CGA[t] - kCC2TC*CGG[t]+ kGC2CC*CGC[t]- kCC2GC*CGG[t]+ kAC2CC*CGT[t]- kCC2AC*CGG[t],
       CGT'[t] == kAT2AC*CAT[t]-kAC2AT*CGT[t] + kAG2AC*CCT[t]- kAC2AG*CGT[t]+ kAA2AC*CTT[t]- kAC2AA*CGT[t]+
     kGA2GT*CGA[t] - kAC2TC*CGT[t]+ kGC2AC*CGC[t]- kAC2GC*CGT[t]+ kCC2AC*CGG[t]- kAC2CC*CGT[t],
       CTA'[t] == kAA2TA*CAA[t]-kTA2AA*CTA[t] + kCA2TA*CCA[t]- kTA2CA*CTA[t]+ kGA2TA*CGA[t]- kTA2GA*CTA[t]+
     kGA2TA*CTC[t] - kTA2GA*CTA[t]+ kAA2TA*CTT[t]- kTA2AA*CTA[t]+ kCA2TA*CTG[t]- kTA2CA*CTA[t],
       CTC'[t] == kAC2TC*CAC[t]-kGA2GT*CTC[t] + kCC2TC*CCC[t]- kGA2GG*CTC[t]+ kGC2GA*CGC[t]- kGA2GC*CTC[t]+
     kTA2GA*CTA[t] - kGA2TA*CTC[t]+ kAA2GA*CTT[t]- kGA2AA*CTC[t]+ kCA2GA*CTG[t]- kGA2CA*CTC[t],
       CTG'[t] == kAG2TG*CAG[t]-kCA2CT*CTG[t] + kCG2CA*CCG[t]- kCA2CG*CTG[t]+ kCC2CA*CGG[t]- kCA2CC*CTG[t]+
     kTA2CA*CTA[t] - kCA2TA*CTG[t]+ kGA2CA*CTC[t]- kCA2GA*CTG[t]+ kAA2CA*CTT[t]- kCA2AA*CTG[t],
       CTT'[t] == kAT2AA*CAT[t]-kAA2AT*CTT[t] + kAG2AA*CCT[t]- kAA2AG*CTT[t]+ kAC2AA*CGT[t]- kAA2AC*CTT[t]+
     kTA2AA*CTA[t] - kAA2TA*CTT[t]+ kGA2AA*CTC[t]- kAA2GA*CTT[t]+ kCA2AA*CTG[t]- kAA2CA*CTT[t],
       CAA+CAC+CAG+CAT+CCA+CCC+CCG+CCT+CGA+CGC+CGG+CGT+CTA+CTC+CTG+CTT==1,
       CAA[0]==CAA0,CAC[0]==CAC0,CAG[0]==CAG0,CAT[0]==CAT0,CCA[0]==CCA0,
       CCC[0]==CCC0,CCG[0]==CCG0,CCT[0]==CCT0,CGA[0]==CGA0,CGC[0]==CGC0,
       CGG[0]==CGG0,CGT[0]==CGT0,CTA[0]==CTA0,CTC[0]==CTC0,CTG[0]==CTG0,CTT[0]==CTT0},
       {CAA,CAC,CAG,CAT,CCA,CCC,CCG,CCT,CGA,CGC,CGG,CGT,CTA,CTC,CTG,CTT}, t]//FullSimplify   *)
```

The equilibrium dimer fractions can therefore be computed via:

```
(*   Solve[
        kCA2AA*CCA - kAA2CA*CAA + kTA2AA*CTA - kAA2TA*CAA + kGA2AA*CGA - kAA2GA*CAA +
      kAC2AA*CAC - kAA2AC*CAA + kAT2AA*CAT - kAA2AT*CAA + kAG2AA*CAG - kAA2AG*CAA==0 &&
        kCC2AC*CCC - kAC2CC*CAC + kGA2GT*CTC - kAC2TC*CAC + kGC2AC*CGC - kAC2GC*CAC +
      kAA2AC*CAA - kAC2AA*CAC + kAT2AC*CAT - kAC2AT*CAC + kAG2AC*CAG - kAC2AG*CAC==0 &&
        kCG2AG*CCG - kAG2CG*CAG + kCA2CT*CTG - kAG2TG*CAG + kCC2CT*CGG - kAG2GG*CAG +
      kAA2AG*CAA - kAG2AA*CAG + kAC2AG*CAC - kAG2AC*CAG + kAT2AG*CAT - kAG2AT*CAG==0 &&
        kAG2AT*CCT - kAT2AG*CAT + kAA2AT*CTT - kAT2AA*CAT + kAC2AT*CGT - kAT2AC*CAT +
      kAA2AT*CAA - kAT2AA*CAT + kAC2AT*CAC - kAT2AC*CAT + kAG2AT*CAG - kAT2AG*CAT==0 &&
        kAA2CA*CAA - kCA2AA*CCA + kTA2CA*CTA - kCA2TA*CCA + kGA2CA*CGA - kCA2GA*CCA +
      kCC2CA*CCC - kCA2CC*CCA + kAG2TG*CCT - kCA2CT*CCA + kCG2CA*CCG - kCA2CG*CCA==0 &&
        kAC2CC*CAC - kCC2AC*CCC + kGA2GG*CTC - kCC2TC*CCC + kGC2CC*CGC - kCC2GC*CCC +
      kCA2CC*CCA - kCC2CA*CCC + kAG2GG*CCT - kCC2CT*CCC + kCG2CC*CCG - kCC2CG*CCC==0 &&
        kAG2CG*CAG - kCG2AG*CCG + kCA2CG*CTG - kCG2CA*CCG + kCC2CG*CGG - kCG2CC*CCG +
      kCA2CG*CCA - kCG2CA*CCG + kCC2CG*CCC - kCG2CC*CCG + kAG2CG*CCT - kCG2AG*CCG==0 &&
        kAT2AG*CAT - kAG2AT*CCT + kAA2AG*CTT - kAG2AA*CCT + kAC2AG*CGT - kAG2AC*CCT +
      kCA2CT*CCA - kAG2TG*CCT + kCC2CT*CCC - kAG2GG*CCT + kCG2AG*CCG - kAG2CG*CCT==0 &&
        kAA2GA*CAA - kGA2AA*CGA + kCA2GA*CCA - kGA2CA*CGA + kTA2GA*CTA - kGA2TA*CGA +
      kGC2GA*CGC - kGA2GC*CGA + kAC2TC*CGT - kGA2GT*CGA + kCC2TC*CGG - kGA2GG*CGA==0 &&
        kAC2GC*CAC - kGC2AC*CGC + kCC2GC*CCC - kGC2CC*CGC + kGA2GC*CTC - kGC2GA*CGC +
      kGA2GC*CGA - kGC2GA*CGC + kAC2GC*CGT - kGC2AC*CGC + kCC2GC*CGG - kGC2CC*CGC==0 &&
        kAG2GG*CAG - kCC2CT*CGG + kCG2CC*CCG - kCC2CG*CGG + kCA2CC*CTG - kCC2CA*CGG +
      kGA2GG*CGA - kCC2TC*CGG + kGC2CC*CGC - kCC2GC*CGG + kAC2CC*CGT - kCC2AC*CGG==0 &&
        kAT2AC*CAT - kAC2AT*CGT + kAG2AC*CCT - kAC2AG*CGT + kAA2AC*CTT - kAC2AA*CGT +
      kGA2GT*CGA - kAC2TC*CGT + kGC2AC*CGC - kAC2GC*CGT + kCC2AC*CGG - kAC2CC*CGT==0 &&
        kAA2TA*CAA - kTA2AA*CTA + kCA2TA*CCA - kTA2CA*CTA + kGA2TA*CGA - kTA2GA*CTA +
      kGA2TA*CTC - kTA2GA*CTA + kAA2TA*CTT - kTA2AA*CTA + kCA2TA*CTG - kTA2CA*CTA==0 &&
        kAC2TC*CAC - kGA2GT*CTC + kCC2TC*CCC - kGA2GG*CTC + kGC2GA*CGC - kGA2GC*CTC +
      kTA2GA*CTA - kGA2TA*CTC + kAA2GA*CTT - kGA2AA*CTC + kCA2GA*CTG - kGA2CA*CTC==0 &&
        kAG2TG*CAG - kCA2CT*CTG + kCG2CA*CCG - kCA2CG*CTG + kCC2CA*CGG - kCA2CC*CTG +
      kTA2CA*CTA - kCA2TA*CTG + kGA2CA*CTC - kCA2GA*CTG + kAA2CA*CTT - kCA2AA*CTG==0 &&
        kAT2AA*CAT - kAA2AT*CTT + kAG2AA*CCT - kAA2AG*CTT + kAC2AA*CGT - kAA2AC*CTT +
      kTA2AA*CTA - kAA2TA*CTT + kGA2AA*CTC - kAA2GA*CTT + kCA2AA*CTG - kAA2CA*CTT==0 &&
        CAA+CAC+CAG+CAT+CCA+CCC+CCG+CCT+CGA+CGC+CGG+CGT+CTA+CTC+CTG+CTT==1,
       {CAA,CAC,CAG,CAT,CCA,CCC,CCG,CCT,CGA,CGC,CGG,CGT,CTA,CTC,CTG,CTT}]//FullSimplify   *)
```

However, the above system of 17 equations with its 48 unique coefficients (unique after accounting for the rate constant equivalencies caused by double-strandedness of DNA) is very heavy for *Mathematica.* Adding different conditions, such as the rate constants being greater than 0, setting the dimer fractions being always greater than 0, restricting the solution region to different boundaries and trying all the matrix methodologies discussed above, does not help in getting solutions with 24GB of RAM and 12 CPUs.

By exploring the solutions for an arbitrary set of numeric inputs, one can, however, see the equivalencies in the pairs of evaluated C_{ij} dimer contents that are reverse complementar to each other. This is exactly what the oligo version of the Chargaff's second parity rule states, which means that the oligo version of the observed species-invariant base-count equivalencies in genomes are also the result of the intrinsic rate constant equalities in the cross-mutation networks. Besides the numeric trials, the equivalency is clear from the comparison of the state equations for the reverse-complementar pairs. As an example, let us consider the state equations for the equilibrium fractions of ApA and its reverse complementar TpT dimers. After a slight reorganisations, the equations for C_{AA} and C_{TT} correspondingly look like:

```
(*   {kCA2AA*CCA-kAA2CA*CAA+kTA2AA*CTA-kAA2TA*CAA+kGA2AA*CGA-kAA2GA*CAA+
   kAC2AA*CAC-kAA2AC*CAA+kAT2AA*CAT-kAA2AT*CAA+kAG2AA*CAG-kAA2AG*CAA==0,
     kCA2AA*CTG-kAA2CA*CTT+kTA2AA*CTA-kAA2TA*CTT+kGA2AA*CTC-kAA2GA*CTT+
   kAC2AA*CGT-kAA2AC*CTT+kAT2AA*CAT-kAA2AT*CTT+kAG2AA*CCT-kAA2AG*CTT==0}    *)
```

As you can see, those equations have exactly the same set of rate constants (stemming from the double-strandedness of DNA as decribed before) and the same values of the expressions (0). Hence, one of the solutions to this pair would be the set where $\{C_{CA}= C_{TG}, C_{AA}= C_{TT}, C_{GA}= C_{TC}, C_{AC}= C_{GT}, C_{AG}= C_{CT}\}$.

Similar equalities can be inferred from the other 5 pairs of equations that have exactly the same set of rate constants (see below for the complete pairings).

```
(*
    (*pair for CAA and CTT*)
 kCA2AA*CCA - kAA2CA*CAA + kTA2AA*CTA - kAA2TA*CAA + kGA2AA*CGA - kAA2GA*CAA +
   kAC2AA*CAC - kAA2AC*CAA + kAT2AA*CAT - kAA2AT*CAA + kAG2AA*CAG - kAA2AG*CAA==0 &&
 kAT2AA*CAT - kAA2AT*CTT + kAG2AA*CCT - kAA2AG*CTT + kAC2AA*CGT - kAA2AC*CTT +
   kTA2AA*CTA - kAA2TA*CTT + kGA2AA*CTC - kAA2GA*CTT + kCA2AA*CTG - kAA2CA*CTT==0

    (*pair for CAC and CGT*)
 kCC2AC*CCC - kAC2CC*CAC + kGA2GT*CTC - kAC2TC*CAC + kGC2AC*CGC - kAC2GC*CAC +
   kAA2AC*CAA - kAC2AA*CAC + kAT2AC*CAT - kAC2AT*CAC + kAG2AC*CAG - kAC2AG*CAC==0 &&
 kAT2AC*CAT - kAC2AT*CGT + kAG2AC*CCT - kAC2AG*CGT + kAA2AC*CTT - kAC2AA*CGT +
   kGA2GT*CGA - kAC2TC*CGT + kGC2AC*CGC - kAC2GC*CGT + kCC2AC*CGG - kAC2CC*CGT==0

    (*pair for CAG and CCT*)
 kCG2AG*CCG - kAG2CG*CAG + kCA2CT*CTG - kAG2TG*CAG + kCC2CT*CGG - kAG2GG*CAG +
   kAA2AG*CAA - kAG2AA*CAG + kAC2AG*CAC - kAG2AC*CAG + kAT2AG*CAT - kAG2AT*CAG==0 &&
 kAT2AG*CAT - kAG2AT*CCT + kAA2AG*CTT - kAG2AA*CCT + kAC2AG*CGT - kAG2AC*CCT +
   kCA2CT*CCA - kAG2TG*CCT + kCC2CT*CCC - kAG2GG*CCT + kCG2AG*CCG - kAG2CG*CCT==0

    (*unpaired CAT*)
 kAG2AT*CCT - kAT2AG*CAT + kAA2AT*CTT - kAT2AA*CAT + kAC2AT*CGT - kAT2AC*CAT +
   kAA2AT*CAA - kAT2AA*CAT + kAC2AT*CAC - kAT2AC*CAT + kAG2AT*CAG - kAT2AG*CAT==0

    (*pair for CCA and CTG*)
 kAA2CA*CAA - kCA2AA*CCA + kTA2CA*CTA - kCA2TA*CCA + kGA2CA*CGA - kCA2GA*CCA +
   kCC2CA*CCC - kCA2CC*CCA + kAG2TG*CCT - kCA2CT*CCA + kCG2CA*CCG - kCA2CG*CCA==0 &&
 kAG2TG*CAG - kCA2CT*CTG + kCG2CA*CCG - kCA2CG*CTG + kCC2CA*CGG - kCA2CC*CTG +
   kTA2CA*CTA - kCA2TA*CTG + kGA2CA*CTC - kCA2GA*CTG + kAA2CA*CTT - kCA2AA*CTG==0

    (*pair for CCC and CGG*)
 kAC2CC*CAC - kCC2AC*CCC + kGA2GG*CTC - kCC2TC*CCC + kGC2CC*CGC - kCC2GC*CCC +
   kCA2CC*CCA - kCC2CA*CCC + kAG2GG*CCT - kCC2CT*CCC + kCG2CC*CCG - kCC2CG*CCC==0 &&
 kAG2GG*CAG - kCC2CT*CGG + kCG2CC*CCG - kCC2CG*CGG + kCA2CC*CTG - kCC2CA*CGG +
   kGA2GG*CGA - kCC2TC*CGG + kGC2CC*CGC - kCC2GC*CGG + kAC2CC*CGT - kCC2AC*CGG==0

    (*unpaired CCG*)
 kAG2CG*CAG - kCG2AG*CCG + kCA2CG*CTG - kCG2CA*CCG + kCC2CG*CGG - kCG2CC*CCG +
   kCA2CG*CCA - kCG2CA*CCG + kCC2CG*CCC - kCG2CC*CCG + kAG2CG*CCT - kCG2AG*CCG==0

    (*pair for CGA and CTC*)
 kAA2GA*CAA - kGA2AA*CGA + kCA2GA*CCA - kGA2CA*CGA + kTA2GA*CTA - kGA2TA*CGA +
   kGC2GA*CGC - kGA2GC*CGA + kAC2TC*CGT - kGA2GT*CGA + kCC2TC*CGG - kGA2GG*CGA==0 &&
 kAC2TC*CAC - kGA2GT*CTC + kCC2TC*CCC - kGA2GG*CTC + kGC2GA*CGC - kGA2GC*CTC +
   kTA2GA*CTA - kGA2TA*CTC + kAA2GA*CTT - kGA2AA*CTC + kCA2GA*CTG - kGA2CA*CTC==0

    (*unpaired CGC*)
 kAC2GC*CAC - kGC2AC*CGC + kCC2GC*CCC - kGC2CC*CGC + kGA2GC*CTC - kGC2GA*CGC +
   kGA2GC*CGA - kGC2GA*CGC + kAC2GC*CGT - kGC2AC*CGC + kCC2GC*CGG - kGC2CC*CGC==0

    (*unpaired CTA*)
 kAA2TA*CAA - kTA2AA*CTA + kCA2TA*CCA - kTA2CA*CTA + kGA2TA*CGA - kTA2GA*CTA +
   kGA2TA*CTC - kTA2GA*CTA + kAA2TA*CTT - kTA2AA*CTA + kCA2TA*CTG - kTA2CA*CTA==0
 *)
```

We can still try to find the solutions by reducing the number of equations through the elimination of one equation in each pair, assuming that the solution is to be searched in the Chargaff's compliance region, which follows from the system directly, as discussed above. The eliminated equations are the ones for C_{TT}, C_{GT}, C_{CT}, C_{TG}, C_{GG} and C_{TC}, as those are equal to their reverse complementar counterparts (added via the With[ ] function below). This reduces the system from 17 to 11 equations.

```
(*   Solve[
  With[{CTT=CAA, CGT=CAC, CCT=CAG, CTG=CCA,CGG=CCC, CTC=CGA},
       {kCA2AA*CCA - kAA2CA*CAA + kTA2AA*CTA - kAA2TA*CAA + kGA2AA*CGA - kAA2GA*CAA +
       kAC2AA*CAC - kAA2AC*CAA + kAT2AA*CAT - kAA2AT*CAA + kAG2AA*CAG - kAA2AG*CAA==0 &&
        kCC2AC*CCC - kAC2CC*CAC + kGA2GT*CTC - kAC2TC*CAC + kGC2AC*CGC - kAC2GC*CAC +
       kAA2AC*CAA - kAC2AA*CAC + kAT2AC*CAT - kAC2AT*CAC + kAG2AC*CAG - kAC2AG*CAC==0 &&
        kCG2AG*CCG - kAG2CG*CAG + kCA2CT*CTG - kAG2TG*CAG + kCC2CT*CGG - kAG2GG*CAG +
       kAA2AG*CAA - kAG2AA*CAG + kAC2AG*CAC - kAG2AC*CAG + kAT2AG*CAT - kAG2AT*CAG==0 &&
        kAG2AT*CCT - kAT2AG*CAT + kAA2AT*CTT - kAT2AA*CAT + kAC2AT*CGT - kAT2AC*CAT +
       kAA2AT*CAA - kAT2AA*CAT + kAC2AT*CAC - kAT2AC*CAT + kAG2AT*CAG - kAT2AG*CAT==0 &&
        kAA2CA*CAA - kCA2AA*CCA + kTA2CA*CTA - kCA2TA*CCA + kGA2CA*CGA - kCA2GA*CCA +
       kCC2CA*CCC - kCA2CC*CCA + kAG2TG*CCT - kCA2CT*CCA + kCG2CA*CCG - kCA2CG*CCA==0 &&
        kAC2CC*CAC - kCC2AC*CCC + kGA2GG*CTC - kCC2TC*CCC + kGC2CC*CGC - kCC2GC*CCC +
       kCA2CC*CCA - kCC2CA*CCC + kAG2GG*CCT - kCC2CT*CCC + kCG2CC*CCG - kCC2CG*CCC==0 &&
        kAG2CG*CAG - kCG2AG*CCG + kCA2CG*CTG - kCG2CA*CCG + kCC2CG*CGG - kCG2CC*CCG +
       kCA2CG*CCA - kCG2CA*CCG + kCC2CG*CCC - kCG2CC*CCG + kAG2CG*CCT - kCG2AG*CCG==0 &&
        kAA2GA*CAA - kGA2AA*CGA + kCA2GA*CCA - kGA2CA*CGA + kTA2GA*CTA - kGA2TA*CGA +
       kGC2GA*CGC - kGA2GC*CGA + kAC2TC*CGT - kGA2GT*CGA + kCC2TC*CGG - kGA2GG*CGA==0 &&
        kAC2GC*CAC - kGC2AC*CGC + kCC2GC*CCC - kGC2CC*CGC + kGA2GC*CTC - kGC2GA*CGC +
       kGA2GC*CGA - kGC2GA*CGC + kAC2GC*CGT - kGC2AC*CGC + kCC2GC*CGG - kGC2CC*CGC==0 &&
        kAA2TA*CAA - kTA2AA*CTA + kCA2TA*CCA - kTA2CA*CTA + kGA2TA*CGA - kTA2GA*CTA +
       kGA2TA*CTC - kTA2GA*CTA + kAA2TA*CTT - kTA2AA*CTA + kCA2TA*CTG - kTA2CA*CTA==0 &&
        CAA+CAC+CAG+CAT+CCA+CCC+CCG+CCT+CGA+CGC+CGG+CGT+CTA+CTC+CTG+CTT== 1}],
    {CAA,CAC,CAG,CAT,CCA,CCC,CCG,CGA,CGC,CTA}]//FullSimplify   *)
```

Unfortunately, none of the approaches to solve the system (just several examples are written below, discussed in Section 2) worked under the existing computational resources. To this end, the solutions of the full hypercube network is to be examined numerically for now.

```
(*   $eq=With[{CTT=CAA, CGT=CAC, CCT=CAG, CTG=CCA,CGG=CCC, CTC=CGA},
      {kCA2AA*CCA - kAA2CA*CAA + kTA2AA*CTA - kAA2TA*CAA + kGA2AA*CGA - kAA2GA*CAA +
      kAC2AA*CAC - kAA2AC*CAA + kAT2AA*CAT - kAA2AT*CAA + kAG2AA*CAG - kAA2AG*CAA==0 &&
       kCC2AC*CCC - kAC2CC*CAC + kGA2GT*CTC - kAC2TC*CAC + kGC2AC*CGC - kAC2GC*CAC +
      kAA2AC*CAA - kAC2AA*CAC + kAT2AC*CAT - kAC2AT*CAC + kAG2AC*CAG - kAC2AG*CAC==0 &&
       kCG2AG*CCG - kAG2CG*CAG + kCA2CT*CTG - kAG2TG*CAG + kCC2CT*CGG - kAG2GG*CAG +
      kAA2AG*CAA - kAG2AA*CAG + kAC2AG*CAC - kAG2AC*CAG + kAT2AG*CAT - kAG2AT*CAG==0 &&
       kAG2AT*CCT - kAT2AG*CAT + kAA2AT*CTT - kAT2AA*CAT + kAC2AT*CGT - kAT2AC*CAT +
      kAA2AT*CAA - kAT2AA*CAT + kAC2AT*CAC - kAT2AC*CAT + kAG2AT*CAG - kAT2AG*CAT==0 &&
       kAA2CA*CAA - kCA2AA*CCA + kTA2CA*CTA - kCA2TA*CCA + kGA2CA*CGA - kCA2GA*CCA +
      kCC2CA*CCC - kCA2CC*CCA + kAG2TG*CCT - kCA2CT*CCA + kCG2CA*CCG - kCA2CG*CCA==0 &&
       kAC2CC*CAC - kCC2AC*CCC + kGA2GG*CTC - kCC2TC*CCC + kGC2CC*CGC - kCC2GC*CCC +
      kCA2CC*CCA - kCC2CA*CCC + kAG2GG*CCT - kCC2CT*CCC + kCG2CC*CCG - kCC2CG*CCC==0 &&
       kAG2CG*CAG - kCG2AG*CCG + kCA2CG*CTG - kCG2CA*CCG + kCC2CG*CGG - kCG2CC*CCG +
      kCA2CG*CCA - kCG2CA*CCG + kCC2CG*CCC - kCG2CC*CCG + kAG2CG*CCT - kCG2AG*CCG==0 &&
       kAA2GA*CAA - kGA2AA*CGA + kCA2GA*CCA - kGA2CA*CGA + kTA2GA*CTA - kGA2TA*CGA +
      kGC2GA*CGC - kGA2GC*CGA + kAC2TC*CGT - kGA2GT*CGA + kCC2TC*CGG - kGA2GG*CGA==0 &&
       kAC2GC*CAC - kGC2AC*CGC + kCC2GC*CCC - kGC2CC*CGC + kGA2GC*CTC - kGC2GA*CGC +
      kGA2GC*CGA - kGC2GA*CGC + kAC2GC*CGT - kGC2AC*CGC + kCC2GC*CGG - kGC2CC*CGC==0 &&
       kAA2TA*CAA - kTA2AA*CTA + kCA2TA*CCA - kTA2CA*CTA + kGA2TA*CGA - kTA2GA*CTA +
      kGA2TA*CTC - kTA2GA*CTA + kAA2TA*CTT - kTA2AA*CTA + kCA2TA*CTG - kTA2CA*CTA==0 &&
       CAA+CAC+CAG+CAT+CCA+CCC+CCG+CCT+CGA+CGC+CGG+CGT+CTA+CTC+CTG+CTT== 1}];
$var = {CAA,CAC,CAG,CAT,CCA,CCC,CCG,CGA,CGC,CTA};
$mx = CoefficientArrays[$eq, $var];   *)
```

```
(*   $res = LinearSolve[ $mx[[2]], -$mx[[1]] ]//FullSimplify   *)
```

```
(*   $res = RowReduce[Transpose[Join[Transpose[$mx[[2]]],{-$mx[[1]]}]]]//
  FullSimplify   *)
```

In summary of the performed steps in this section, we have extended the cross-mutation network to represent the di-nucleotide (dimer) fractions in genomes. The set of rate constants in this extension are now accounting for the neighbouring nucleotide effect. However, we demonstrated that the rate constant symmetries, analogous to the single-base case, apply to the oligomeric extension as well. We then wrote down the system of differential equations via the reduced 48 coefficients (mutation rate constants). We showed that the system is difficult to solve symbolically, however, its structure implies that a solution where the equilibrium fractions (counts) of reverse complementar dimers are equal (oligo-version of the Chargaff's rule) is a conforming solution for the system. We then go on using this property to reduce the system into just 11 equations, which is still out of the power of *Mathematica* under the constraints of our resources.

4. The Case of Dimers in a Double-Stranded Genome Assuming an Absence of Sequence-Context Dependence of Mutation Rates

If we assume that there is no neighbouring nucleotide effect on the mutation rates, we can use the same i, j, k, l, m, n rate constants (see Section 2) to describe all the cross mutations in the hypercube-based cross-mutation model. Of course, this assumption is crude for especially some dimers, where the neighbouring effect can be substantial (for instance Cs and Gs in CpG dimers have substantially higher substitution rates). However, this relatively reduced dimeric model could still be useful to decribe the overall dimeric contents in genomes and to outline the most severe neighbour effects on mutation rates. The system of dimer-state equations, constructed in Section 3, reduces here into the following:

```
(*  DSolve[
     {CAA'[t]==(CAC[t]+CCA[t])*i+(CAT[t]+CTA[t])*l+(CAG[t]+CGA[t])*m-2*CAA[t]*(j+l+n),
      CAC'[t]==CCC[t]*i+CAA[t]*j+CAG[t]*k+CTC[t]*l+CGC[t]*m+CAT[t]*n-CAC[t]*(i+j+k+l+m+n),
      CAG'[t]==CCG[t]*i+CAT[t]*j+CAC[t]*k+CTG[t]*l+CGG[t]*m+CAA[t]*n-CAG[t]*(i+j+k+l+m+n),
      CAT'[t]==(CAG[t]+CCT[t])*i+(CAA[t]+CTT[t])*l+(CAC[t]+CGT[t])*m-2*CAT[t]*(j+l+n),
      CCA'[t]==CCC[t]*i+CAA[t]*j+CGA[t]*k+CCT[t]*l+CCG[t]*m+CTA[t]*n-CCA[t]*(i+j+k+l+m+n),
      CCC'[t]==(CAC[t]+CCA[t])*j+(CCG[t]+CGC[t])*k+(CCT[t]+CTC[t])*n-2*CCC[t]*(i+k+m),
      CCG'[t]==(CAG[t]+CCT[t])*j+(CCC[t]+CGG[t])*k+(CCA[t]+CTG[t])*n-2*CCG[t]*(i+k+m),
      CCT'[t]==CCG[t]*i+CAT[t]*j+CGT[t]*k+CCA[t]*l+CCC[t]*m+CTT[t]*n-CCT[t]*(i+j+k+l+m+n),
      CGA'[t]==CGC[t]*i+CTA[t]*j+CCA[t]*k+CGT[t]*l+CGG[t]*m+CAA[t]*n-CGA[t]*(i+j+k+l+m+n),
      CGC'[t]==(CGA[t]+CTC[t])*j+(CCC[t]+CGG[t])*k+(CAC[t]+CGT[t])*n-2*CGC[t]*(i+k+m),
      CGG'[t]==(CGT[t]+CTG[t])*j+(CCG[t]+CGC[t])*k+(CAG[t]+CGA[t])*n-2*CGG[t]*(i+k+m),
      CGT'[t]==CGG[t]*i+CTT[t]*j+CCT[t]*k+CGA[t]*l+CGC[t]*m+CAT[t]*n-CGT[t]*(i+j+k+l+m+n),
      CTA'[t]==(CGA[t]+CTC[t])*i+(CAA[t]+CTT[t])*l+(CCA[t]+CTG[t])*m-2*CTA[t]*(j+l+n),
      CTC'[t]==CGC[t]*i+CTA[t]*j+CTG[t]*k+CAC[t]*l+CCC[t]*m+CTT[t]*n-CTC[t]*(i+j+k+l+m+n),
      CTG'[t]==CGG[t]*i+CTT[t]*j+CTC[t]*k+CAG[t]*l+CCG[t]*m+CTA[t]*n-CTG[t]*(i+j+k+l+m+n),
      CTT'[t]==(CGT[t]+CTG[t])*i+(CAT[t]+CTA[t])*l+(CCT[t]+CTC[t])*m-2*CTT[t]*(j+l+n)},
      {CAA,CAC,CAG,CAT,CCA,CCC,CCG,CCT,CGA,CGC,CGG,CGT,CTA,CTC,CTG,CTT}, t] //
 FullSimplify   *)
```

At equilibrium:

```
eq4 = {
    i*CCA - j*CAA + l*CTA - l*CAA + m*CGA - n*CAA + i*CAC - j*CAA + l*CAT - l*CAA + m*CAG - n*CAA == 0,
    i*CCC - j*CAC + l*CTC - l*CAC + m*CGC - n*CAC + j*CAA - i*CAC + n*CAT - m*CAC + k*CAG - k*CAC == 0,
    i*CCG - j*CAG + l*CTG - l*CAG + m*CGG - n*CAG + n*CAA - m*CAG + k*CAC - k*CAG + j*CAT - i*CAG == 0,
    i*CCT - j*CAT + l*CTT - l*CAT + m*CGT - n*CAT + l*CAA - l*CAT + m*CAC - n*CAT + i*CAG - j*CAT == 0,
    j*CAA - i*CCA + n*CTA - m*CCA + k*CGA - k*CCA + i*CCC - j*CCA + l*CCT - l*CCA + m*CCG - n*CCA == 0,
    j*CAC - i*CCC + n*CTC - m*CCC + k*CGC - k*CCC + j*CCA - i*CCC + n*CCT - m*CCC + k*CCG - k*CCC == 0,
    j*CAG - i*CCG + n*CTG - m*CCG + k*CGG - k*CCG + n*CCA - m*CCG + k*CCC - k*CCG + j*CCT - i*CCG == 0,
    j*CAT - i*CCT + n*CTT - m*CCT + k*CGT - k*CCT + l*CCA - l*CCT + m*CCC - n*CCT + i*CCG - j*CCT == 0,
    n*CAA - m*CGA + k*CCA - k*CGA + j*CTA - i*CGA + i*CGC - j*CGA + l*CGT - l*CGA + m*CGG - n*CGA == 0,
    n*CAC - m*CGC + k*CCC - k*CGC + j*CTC - i*CGC + j*CGA - i*CGC + n*CGT - m*CGC + k*CGG - k*CGC == 0,
    n*CAG - m*CGG + k*CCG - k*CGG + j*CTG - i*CGG + n*CGA - m*CGG + k*CGC - k*CGG + j*CGT - i*CGG == 0,
    n*CAT - m*CGT + k*CCT - k*CGT + j*CTT - i*CGT + l*CGA - l*CGT + m*CGC - n*CGT + i*CGG - j*CGT == 0,
    l*CAA - l*CTA + m*CCA - n*CTA + i*CGA - j*CTA + i*CTC - j*CTA + l*CTT - l*CTA + m*CTG - n*CTA == 0,
    l*CAC - l*CTC + m*CCC - n*CTC + i*CGC - j*CTC + j*CTA - i*CTC + n*CTT - m*CTC + k*CTG - k*CTC == 0,
    l*CAG - l*CTG + m*CCG - n*CTG + i*CGG - j*CTG + n*CTA - m*CTG + k*CTC - k*CTG + j*CTT - i*CTG == 0,
    l*CAT - l*CTT + m*CCT - n*CTT + i*CGT - j*CTT + l*CTA - l*CTT + m*CTC - n*CTT + i*CTG - j*CTT == 0,
    CAA + CAC + CAG + CAT + CCA + CCC + CCG + CCT + CGA + CGC + CGG + CGT + CTA + CTC + CTG + CTT == 1
  };
var4 = {CAA, CAC, CAG, CAT, CCA, CCC, CCG, CCT, CGA, CGC, CGG, CGT, CTA, CTC, CTG, CTT};
```

This system is still complex for *Mathematica* (when trying all the methods described in Section 2), but we managed to find exact symbolic solutions by carefully exploring the emergent groups in the equations. First, let us just assign an arbitrary set of numerical values to the {i, j, k, l, m, n} rate constants.

```
{i, j, k, l, m, n} = {1.2, 2.3, 1.5, 4.1, 3.8, 5.3};
```

We can now find the numerical solutions:

```
NSolve[eq4, var4]
```

```
{{CAA → 0.0393676, CAC → 0.0598388, CAG → 0.0598388, CAT → 0.0393676,
  CCA → 0.0598388, CCC → 0.0909549, CCG → 0.0909549, CCT → 0.0598388,
  CGA → 0.0598388, CGC → 0.0909549, CGG → 0.0909549, CGT → 0.0598388,
  CTA → 0.0393676, CTC → 0.0598388, CTG → 0.0598388, CTT → 0.0393676}}
```

```
Clear[i, j, k, l, m, n]
```

The numerical solution to the system with the arbitrary assigned coefficients implies that

$(C_{AA}= C_{TT}) = (C_{AT}) = (C_{TA}) = a,$ # 0.0393676 in an arbitrary case
$(C_{AC}= C_{GT}) = (C_{AG}= C_{CT}) = (C_{CA}= C_{TG}) = (C_{GA}= C_{TC}) = b,$ # 0.0598388 in an arbitrary case
$(C_{CC}= C_{GG}) = (C_{CG}) = (C_{GC}) = c$ # 0.0909549 in an arbitrary case

where, the brackets denote the equalities expected from the oligo version of the second parity rule. Hence, the empirical rule is, again, emergent from the solution of the system. We just have some additional equalities that are present under the assumption of having no neighbour effect on the mutation rates. Therefore, for such hypothetical genomes, Chargaff could observe even more equalities, and the dimeric composition of the whole genome could have been described by just 3 values, a, b and c.

Let us go on and use the found groups (invariant to the i, j, k, l, m, n numeric values, as soon as those are greater than 0) to further trim the system and get the symbolic solutions for a, b and c. The trimming of the system can be done as described in Section 3, where we take only one equation from each of the found three groups. The solutions to this model will be valuable as we can further compare the outcomes for real genomes with the actual dimeric contents, in which case the distortions from the theoretical values will reveal the presence of strong neighbouring effects on the mutation rates.

We shall solve the reduced system by first creating the sparse arrays and solving the system of linear equations in a matrix convention. The solutions will be for the remaining C_{AA}, C_{AC}, and C_{CC} dimer fractions that correspond to a, b and c above, representing the full solution to the complete system in eq4.

```
eq4trim = With[{CTT = CAA, CAT = CAA, CTA = CAA, CGT = CAC, CAG = CAC, CCT = CAC,
           CCA = CAC, CTG = CAC, CGA = CAC, CTC = CAC, CGG = CCC, CCG = CCC, CGC = CCC},
        {i*CCA - j*CAA + l*CTA - l*CAA + m*CGA - n*CAA + i*CAC - j*CAA + l*CAT - l*CAA + m*CAG - n*CAA == 0,
         i*CCC - j*CAC + l*CTC - l*CAC + m*CGC - n*CAC + j*CAA - i*CAC + n*CAT - m*CAC + k*CAG - k*CAC == 0,
         j*CAC - i*CCC + n*CTC - m*CCC + k*CGC - k*CCC + j*CCA - i*CCC + n*CCT - m*CCC + k*CCG - k*CCC == 0,
         CAA + CAC + CAG + CAT + CCA + CCC + CCG + CCT + CGA + CGC + CGG + CGT + CTA + CTC + CTG + CTT == 1}
          ];
var4trim = {CAA, CAC, CCC};
mx4trim = CoefficientArrays[eq4trim, var4trim]
```

```
{SparseArray[ Specified elements : 1
              Dimensions : {4} ], SparseArray[ Specified elements : 10
                                               Dimensions : {4, 3} ]}
```

```
{MatrixForm[mx4trim [[2]]], MatrixForm[-mx4trim [[1]]] }
```

$$\left\{\begin{pmatrix} -2j-2n & 2i+2m & 0 \\ j+n & -i-j-m-n & i+m \\ 0 & 2j+2n & -2i-2m \\ 4 & 8 & 4 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}\right\}$$

```
LinearSolve[ mx4trim [[2]], -mx4trim [[1]] ] // FullSimplify
```

$$\left\{\frac{(i+m)^2}{4\,(i+j+m+n)^2}, \frac{(i+m)\,(j+n)}{4\,(i+j+m+n)^2}, \frac{(j+n)^2}{4\,(i+j+m+n)^2}\right\}$$

The unique {a, b, c} solution is found and can be easily verified by cross checking against the numerical solutions for any arbitrary set of non-0 and positive {i, j, k, l, m, n} rate constants.

```
{i, j, k, l, m, n} = {1.2, 2.3, 1.5, 4.1, 3.8, 5.3};
```

$$\left\{\frac{(i+m)^2}{4\,(i+j+m+n)^2}, \frac{(i+m)\,(j+n)}{4\,(i+j+m+n)^2}, \frac{(j+n)^2}{4\,(i+j+m+n)^2}\right\}$$

```
{0.0393676, 0.0598388, 0.0909549}
```

```
(* The numbers match with the values obtained numerically with NSolve[ ] above *)
```

```
Clear[i, j, k, l, m, n]
```

5. The Application of the Discussed Solutions to Analyse the Chimpanzee Genome

In {BMC Genomics, 7:316, 2006}, the authors used genome-wide dSNP data from the chimpanzee genome to infer different substitution fractions. The ancestral states of the sites were deduced by comparing the dSNP containing sequences to the homologous ones in humans. The work also reported the normalised substitution fractions to account for the biased occurence of used dSNPs at the sites with different base contents.

Below we shall take the reported normalised values, calculate the overall equilibrium base contents (both monomeric and dimeric) in the chimpanzee genome, and compare those with the actual genomic data.

The dSNP-sequence-context-normalised substitution fractions, in %, for the chimpanzee genome (from the described publication) are:

```
{fAG, fTC, fGA, fCT, fAC, fTG, fGT, fCA, fAT, fTA, fGC, fCG} =
  {12.8, 12.7, 20.9, 21.1, 3.4, 3.4, 4.8, 4.9, 2.8, 2.8, 5.3, 5.2};
```

where f_{ij} denotes the fraction of i →j base (single nucleotide) substitutions. The substitution fractions are proportional to the corresponding mutation rate constants, with the divergence time as the proportionality coefficient. We can thus use these fractions as a replacement for the {i, j, k, l, m, n} rate constants. First, we can average the already negligible (a proof of the correctness of our symmetry assumption) differences between the reported substitution fractions, which are supposed to be equal by the rate constant equality described in Section 2.

$k_{CA}=k_{GT}=i$
$k_{AC}=k_{TG}=j$
$k_{CG}=k_{GC}=k$
$k_{AT}=k_{TA}=l$
$k_{CT}=k_{GA}=m$
$k_{AG}=k_{TC}=n$

```
i = (fCA + fGT) / 2;
j = (fAC + fTG) / 2;
k = (fCG + fGC) / 2;
l = (fAT + fTA) / 2;
m = (fCT + fGA) / 2;
n = (fAG + fTC) / 2;
```

Now we can calculate the equilibrium base contents of the chimpanzee genome using the cross-mutation network solutions, where $\{C_A, C_G, C_T, C_C\}$ are the individual base contents, {a, b, c} are the dimeric contents under the assumption of neighbour-invariant rate constants [where $(C_{AA} = C_{TT}) = (C_{AT}) = (C_{TA}) = a$, $(C_{AC} = C_{GT}) = (C_{AG} = C_{CT}) = (C_{CA} = C_{TG}) = (C_{GA} = C_{TC}) = b$, $(C_{CC} = C_{GG}) = (C_{CG}) = (C_{GC}) = c$].

$$\left\{CA \to \frac{i+m}{2\,(i+j+m+n)},\ CG \to \frac{j+n}{2\,(i+j+m+n)},\ CT \to \frac{i+m}{2\,(i+j+m+n)},\ CC \to \frac{j+n}{2\,(i+j+m+n)}\right\}$$

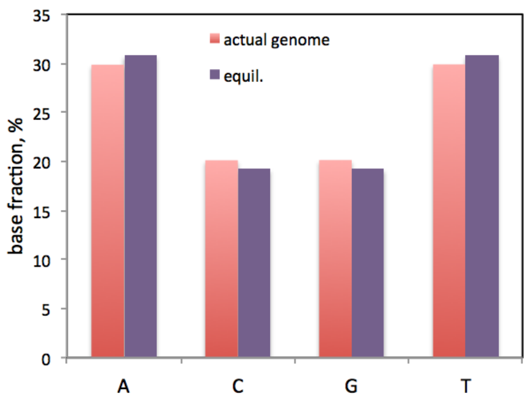
```
{CA → 0.307738, CG → 0.192262, CT → 0.307738, CC → 0.192262}
```

$$\left\{a \to \frac{(i+m)^2}{4\,(i+j+m+n)^2},\ b \to \frac{(i+m)\,(j+n)}{4\,(i+j+m+n)^2},\ c \to \frac{(j+n)^2}{4\,(i+j+m+n)^2}\right\}$$

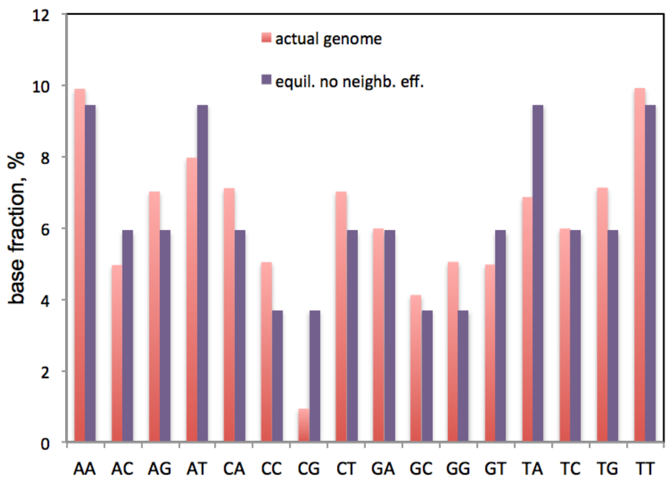
```
{a → 0.0947027, b → 0.0591663, c → 0.0369646}
```

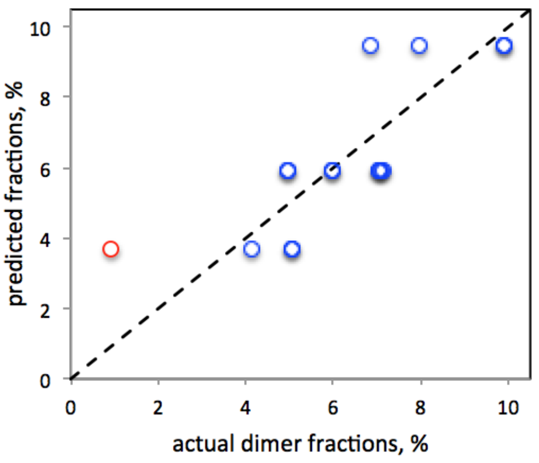
Unfortunately, we cannot apply the full hypercube model (without the neighbour invariance), which could have been applied numerically with NSolve[ ] if we would ave the individual substitution fractions coming from 48 different dimer contexts.

Below you can see the excellent match of the above solutions to the actual data from the chimpanzee genome. When considering the overall base contents (model of Section 2), the match with the experimental base content is excellent. Furthermore, the slight differences are a bit reduced A and T contents and a bit elevated G and C contents in the actual genome. Since our solution is for the equilibrium genome, this would mean that the chimpanzee genome is slightly off equilibrium and will move towards it (do not mix with Chargaff equilibrium) by having more G:C|C:G → A:T|T:A mutations, which is what is inferred in the above quoted publication by other means.



As for comparing the predicted dinucleotide contents to the actual genomic data, please note once more, that since we do not have the substitution fractions for the 48 independent dimeric sequences to use those for the complete rate constant inference, we have only used the solution to the hypercube model with the assumption of zero-neighbouring-nucleotide-effect on the mutation rates (Section 4) by using the 6 rate constants inferred from the single-nucleotide-based substitution data. The comparison of the predicted and actual dimeric contents in the chimpanzee genome is presented below.





The agreement is still rather impressive (see the correlation plot), where, as expected, the major error is for the CpG dimer, where the neighbour effect is the most severe owing to the epigenetic processes recognising CpG sites and accelerating the mutations through epigenetic modifications.