# Colors

**Colormodels, Conversions and Color Difference in Formulas and Code**

Johannes Bildstein

November 24, 2013

## Contents

# Part I.
# Introduction

This article contains a large amount of color conversion formulas and code. You are free to use the code in this article. First the colormodels and conversions are listed, then color difference calculations are listed and at the end there are a few tables with useful data.

You can find a C# library (GPL3 license) with all of the listed colors and some more here:
`www.codeproject.com/Articles/613798/Colorspaces-and-Conversions`

A general note to the code snipplets:
They are written in C#, but since I didn't use any fancy language specific stuff, it should be easily portable to other languages. The code is partially optimized already but still kept readable. Therefore the code might not match the formulas one by one. If you need faster code, you'll have to optimize it yourself.
A few tips on optimizations:

- Don't declare variables within the conversion method but outside (if you use multithreading you might need to take some precautions though)
- Instead of Math.Pow(x, 2) make a simple method that returns $x * x$ (also for $x^3$, make a routine that returns $x * x * x$)
- If possible, make constants for calculations that won't change. Like $\frac{\pi}{180}$ and similar.
- If a part of a formula is used more than one time, only calculate this part once and use the result for all formulas.
- Pass as few variables as possible when calling methods
- Precalculate what's possible to precalculate. Chromatic adaption matrices for example.

You also might want to make a border check after the calculation, e.g. if a number is bigger than 1 or smaller than 0. Or in case of cylindric models you might have to add or subtract 360 to the value.

# Part II.
# Chromatic Adaption

If you convert between colors with a different reference white you have to perform chromatic adaption as follows:

- XYZ values are given in the range [0,1]
- AM is the Adaption Matrix and can be found at section 24
- AM_1 is the inverse Adaption Matrix and can be found at section 24
- Whitepoint.Values is simply X,Y and Z from the Whitepoint in an double array like: double[] Values = new double[3] X, Y, Z ;
- $X_{WS}, Y_{WS}, Z_{WS}$ are the values from the source whitepoint, which you can find at section 23
- $X_{WD}, Y_{WD}, Z_{WD}$ are the values from the destination whitepoint, which you can find at section 23

$$\begin{bmatrix} X_D \\ Y_D \\ Z_D \end{bmatrix} = [M] * \begin{bmatrix} X_S \\ Y_S \\ Z_S \end{bmatrix} \tag{1}$$

Where:

$$[M] = [M_A]^{-1} * \begin{bmatrix} \frac{\rho_D}{\rho_S} & 0 & 0 \\ 0 & \frac{\gamma_D}{\gamma_S} & 0 \\ 0 & 0 & \frac{\beta_D}{\beta_S} \end{bmatrix} * [M_A] \tag{2}$$

And:

$$\begin{bmatrix} \rho_S \\ \gamma_S \\ \beta_S \end{bmatrix} = [M_A] * \begin{bmatrix} X_{WS} \\ Y_{WS} \\ Z_{WS} \end{bmatrix} \tag{3}$$

And:

$$\begin{bmatrix} \rho_D \\ \gamma_D \\ \beta_D \end{bmatrix} = [M_A] * \begin{bmatrix} X_{WD} \\ Y_{WD} \\ Z_{WD} \end{bmatrix} \tag{4}$$

```
ColorXYZ ChromaticAdaption(ColorXYZ XYZ_In, Whitepoint InputWhitepoint,
    Whitepoint OutWhitepoint)
{
    ColorXYZ XYZ_Out = new ColorXYZ();
    double[] S = MultiplyMatrix(AM, InputWhitepoint.Values);
    double[] D = MultiplyMatrix(AM, OutWhitepoint.Values);
    double[,] M = new double[,] { { D[0] / S[0], 0, 0 }, { 0, D[1] / S
        [1], 0 }, { 0, 0, D[2] / S[2] } };
    double[] tmp = MultiplyMatrix(MultiplyMatrix(MultiplyMatrix(AM_1, M
        ), AM), XYZ_In); }
    XYZ_Out.X = tmp[0];
```

```
    XYZ_Out.Y = tmp[1];
    XYZ_Out.Z = tmp[2];

    return XYZ_Out;
}
```

Where:

```
double[] MultiplyMatrix(double[,] a, double[] b)
{
    double[] c = new double[3];
    c[0] = b[0] * a[0, 0] + b[1] * a[0, 1] + b[2] * a[0, 2];
    c[1] = b[0] * a[1, 0] + b[1] * a[1, 1] + b[2] * a[1, 2];
    c[2] = b[0] * a[2, 0] + b[1] * a[2, 1] + b[2] * a[2, 2];
    return c;
}
```

And:

```
double[,] MultiplyMatrix(double[,] a, double[,] b)
{
    double[,] output = new double[3, 3];
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            for (int k = 0; k < 3; k++) { output[i, j] += a[i, k] * b[k
                , j]; }
        }
    }
    return output;
}
```

# Part III.
# Colormodels

## 1. XYZ

The CIE 1931 XYZ colormodel is the basis for all following models, therefore no conversion formulas are needed here.
**Range:**

- X: 0 to 1
- Y: 0 to 1
- Z: 0 to 1

## 2. Yxy

**Range:**

- Y: 0 to 1
- x: not specified
- y: not specified

### 2.1. XYZ to Yxy

- XYZ values are given in the range [0,1]
- Yxy.Y value will be in the range [0,1]
- in case of X+Y+Z=0, x and y should be set to the chromaticity coordinates from the reference white. (see section 23 for data)

$$Y = Y \tag{5}$$

$$x = \frac{X}{X + Y * Z} \tag{6}$$

$$y = \frac{Y}{X + Y + Z} \tag{7}$$

```
ColorYxy XYZToYxy(ColorXYZ XYZ, Whitepoint ReferenceWhite)
{
    ColorYxy Yxy = new ColorYxy();
    Yxy.Y = XYZ.Y;
    if (XYZ.X + XYZ.Y + XYZ.Z == 0)
    {
        Yxy.x = ReferenceWhite.Cx;
```

```
        Yxy.y = ReferenceWhite.Cy;
    }
    else
    {
        Yxy.x = X / (XYZ.X + XYZ.Y + XYZ.Z);
        Yxy.y = Y / (XYZ.X + XYZ.Y + XYZ.Z);
    }
    return Yxy;
}
```

## 2.2. Yxy to XYZ

- Yxy.Y value is given in the range [0,1]
- XYZ values will be in the range [0,1]
- in case of y=0, you should set X=Y=Z=0

$$Y = Y \tag{8}$$

$$X = \frac{x * Y}{y} \tag{9}$$

$$Z = \frac{(1 - x - y) * Y}{y} \tag{10}$$

```
ColorXYZ YxyToXYZ(ColorYxy Yxy, Whitepoint ReferenceWhite)
{
    ColorXYZ XYZ = new ColorXYZ();
    XYZ.Y = Yxy.Y;
    if (Yxy.y == 0)
    {
        XYZ.X = 0;
        XYZ.Y = 0;
        XYZ.Z = 0;
    }
    else
    {
        XYZ.X = (Yxy.x * Yxy.Y) / Yxy.y;
        XYZ.Z = ((1 - Yxy.x - Yxy.y) * Yxy.Y) / Yxy.y;
    }
    return XYZ;
}
```

# 3. Lab

CIE 1976 L*a*b*
**Range:**

- L: 0 to 100
- a: not specified
- b: not specified

## 3.1. XYZ to Lab

- XYZ values are given in the range [0,1]
- Lab.L value will be in the range [0,100]
- $X_r$, $Y_r$, $Z_r$ are the values from the reference white. (see section 23 for data)
- $\varepsilon = 0.008856 = 216/24389$ (Epsilon in code)
- $\kappa = 903.3 = 24389/27$ (Kappa in code)

$$L = 116 * f_y - 16 \tag{11}$$

$$a = 500 * (f_x - f_y) \tag{12}$$

$$b = 200 * (f_y - f_z) \tag{13}$$

Where $f_{x,y,z}$ (replace c with x,y,z):

$$f_c = \begin{cases} \sqrt[3]{c_r} & c_r > \varepsilon \\ \dfrac{\kappa * c_r + 16}{116} & c_r \leq \varepsilon \end{cases} \tag{14}$$

And (replace c with x,y,z and C with X,Y,Z):

$$c_r = \frac{C}{C_r} \tag{15}$$

```
ColorLab XYZToLab(ColorXYZ XYZ, Whitepoint ReferenceWhite)
{
    ColorLab Lab = new ColorLab();
    double y_r = XYZ.Y / ReferenceWhite.Y;
    Lab.L = 116 * Fn_XYZToLab(y_r) - 16;
    Lab.a = 500 * (Fn_XYZToLab(XYZ.X / ReferenceWhite.X) - Fn_XYZToLab(
        y_r));
    Lab.b = 200 * (Fn_XYZToLab(y_r) - Fn_XYZToLab(XYZ.Z /
        ReferenceWhite.Z));

    return Lab;
}
```

With:

```
double Fn_XYZToLab(double val)
{
    if (val <= Epsilon) return ((Kappa * val) + 16d) / 116d;
    else return Math.Pow(val, 1/3d);
}
```

## 3.2. Lab to XYZ

- Lab.L value is given in the range [0,100]
- XYZ values will be in the range [0,1]
- $X_r$, $Y_r$, $Z_r$ are the values from the reference white. (see section 23 for data)
- $\varepsilon = 0.008856 = 216/24389$ (Epsilon in code)
- $\kappa = 903.3 = 24389/27$ (Kappa in code)

$$X = x_r * X_r \tag{16}$$

$$Y = y_r * Y_r \tag{17}$$

$$Z = z_r * Z_r \tag{18}$$

Where:

$$x_r = \begin{cases} f_x^3 & f_x^3 > \varepsilon \\ \dfrac{166 * f_x - 16}{\kappa} & f_x^3 \leq \varepsilon \end{cases} \tag{19}$$

$$y_r = \begin{cases} \left(\dfrac{L+16}{116}\right)^3 & L > \varepsilon * \kappa \\ \dfrac{L}{\kappa} & L \leq \varepsilon * \kappa \end{cases} \tag{20}$$

$$x_r = \begin{cases} f_z^3 & f_z^3 > \varepsilon \\ \dfrac{166 * f_z - 16}{\kappa} & f_z^3 \leq \varepsilon \end{cases} \tag{21}$$

And:

$$f_x = \frac{a}{500} + f_y \tag{22}$$

$$f_z = f_y - \frac{b}{200} \tag{23}$$

$$f_y = \frac{L+16}{116} \tag{24}$$

```
ColorXYZ LabToXYZ(ColorLab Lab, Whitepoint ReferenceWhite)
{
    ColorXYZ XYZ = new ColorXYZ();
    double f_x = Fx_LabToXYZ(Lab.a, Lab.L);
    double f_z = Fz_LabToXYZ(Lab.b, Lab.L);

    double f_x3 = Math.Pow(f_x, 3);
    double x_r;
    if (f_x3 > Epsilon) x_r = f_x3;
    else x_r = ((116d * f_x) - 16d) / Kappa;

    double y_r;
    if (Lab.L > Kappa * Epsilon) y_r = Math.Pow((Lab.L + 16d) / 116d,
       3)
    else y_r = Lab.L / Kappa;

    double f_z3 = Math.Pow(f_z, 3);
    double z_r;
    if (f_z3 > Epsilon) z_r = f_z3;
    else z_r = ((116d * f_z) - 16d) / Kappa;

    XYZ.X = x_r * ReferenceWhite.X;
    XYZ.Y = y_r * ReferenceWhite.Y;
    XYZ.Z = z_r * ReferenceWhite.Z;

    return XYZ;
}
```

With:

```
double Fx_LabToXYZ(double a, double L)
{
    return (a / 500d) + Fy_LabToXYZ(L);
}
```

```
double Fy_LabToXYZ(double L)
{
    return (L + 16) / 116d;
}
```

```
double Fz_LabToXYZ(double b, double L)
{
    return Fy_LabToXYZ(L) - (b / 200d);
}
```

# 4. Luv

CIE 1976 L*u*v*
**Range:**

- L: 0 to 100
- u: not specified
- v: not specified

## 4.1. XYZ to Luv

- XYZ values are given in the range [0,1]
- Luv.L value will be in the range [0,100]
- $X_r$, $Y_r$, $Z_r$ are the values from the reference white. (see section 23 for data)
- $\varepsilon = 0.008856 = 216/24389$ (Epsilon in code)
- $\kappa = 903.3 = 24389/27$ (Kappa in code)

$$L = \begin{cases} 116 * \sqrt[3]{y_r} - 16 & y_r > \varepsilon \\ \kappa * y_r & y_r \leq \varepsilon \end{cases} \tag{25}$$

$$u = 13 * L * (u' - u'_r) \tag{26}$$

$$v = 13 * L * (v' - v'_r) \tag{27}$$

Where:

$$y_r = \frac{Y}{Y_r} \tag{28}$$

And (same for $u_r$' and $u_r$', just use $X_r$,$Y_r$,$Z_r$):

$$u' = \frac{4 * X}{X + 15 * Y + 3 * Z} \tag{29}$$

$$v' = \frac{9 * Y}{X + 15 * Y + 3 * Z} \tag{30}$$

```
ColorLuv XYZToLuv(ColorXYZ XYZ, Whitepoint ReferenceWhite)
{
    ColorLuv Luv = new ColorLuv();
    double y_r = XYZ.Y / ReferenceWhite.Y;
    double u_ = (4 * XYZ.X) / (XYZ.X + 15 * XYZ.Y + 3 * XYZ.Z);
    double v_ = (9 * XYZ.Y) / (XYZ.X + 15 * XYZ.Y + 3 * XYZ.Z);
    double u_r = (4 * ReferenceWhite.X) / (ReferenceWhite.X + 15 *
        ReferenceWhite.Y + 3 * ReferenceWhite.Z);
```

```
    double v_r = (9 * ReferenceWhite.Y) / (ReferenceWhite.X + 15 *
        ReferenceWhite.Y + 3 * ReferenceWhite.Z);

    if (y_r > Epsilon) Luv.L = (116d * Math.Pow(y_r, 1/3d)) - 16d;
    else Luv.L = Kappa * y_r;
    Luv.u = 13 * OutArr[0] * (u_ - u_r);
    Luv.v = 13 * OutArr[0] * (v_ - v_r);

    return Luv;
}
```

## 4.2. Luv to XYZ

- Luv.L value is given in the range [0,100]
- XYZ values will be in the range [0,1]
- $X_r$, $Y_r$, $Z_r$ are the values from the reference white. (see section 23 for data)
- $\varepsilon = 0.008856 = 216/24389$ (Epsilon in code)
- $\kappa = 903.3 = 24389/27$ (Kappa in code)

$$X = \frac{c - b}{a + \frac{1}{3}} \tag{31}$$

$$Y = \begin{cases} \left(\frac{L+16}{116}\right)^3 & L > \kappa * \varepsilon \\ \frac{L}{\kappa} & L \leq \kappa * \varepsilon \end{cases} \tag{32}$$

$$Z = X * a + b \tag{33}$$

Where:

$$a = \frac{1}{3} * \left(\frac{52 * L}{u + 13 * L * u_0} - 1\right) \tag{34}$$

$$b = -5 * Y \tag{35}$$

$$c = Y * \left(\frac{39 * L}{v + 13 * L * v_0} - 5\right) \tag{36}$$

$$u_0 = \frac{4 * X_r}{X_r + 15 * Y_r + 3 * Z_r} \tag{37}$$

$$v_0 = \frac{9 * Y_r}{X_r + 15 * Y_r + 3 * Z_r} \tag{38}$$

```
ColorXYZ LuvToXYZ(ColorLuv Luv, Whitepoint ReferenceWhite)
{
    ColorXYZ XYZ = new ColorXYZ();
    if (Luv.L > Kappa * Epsilon) XYZ.Y = Math.Pow((InArr[0] + 16) / 116
        d, 3);
    else XYZ.Y = Luv.L / Kappa;
    double u0 = (4 * ReferenceWhite.X) / (ReferenceWhite.X + 15 *
        ReferenceWhite.Y + 3 * ReferenceWhite.Z);
    double v0 = (9 * ReferenceWhite.Y) / (ReferenceWhite.X + 15 *
        ReferenceWhite.Y + 3 * ReferenceWhite.Z);
    double a = (((52 * Luv.L) / (Luv.u + 13 * Luv.L * u0)) - 1) / 3d;
    double b = -5 * XYZ.Y;
    double c = (((39 * Luv.L) / (Luv.v + 13 * Luv.L * v0)) - 5) * XYZ.Y
        ;
    XYZ.X = (c - b) / (a + 1/3d);
    XYZ.Z = XYZ.X * a + b;

    return XYZ;
}
```

# 5. RGB

**Range:**

- R: 0 to 1
- G: 0 to 1
- B: 0 to 1

## 5.1. XYZ to RGB

- XYZ values are given in the range [0,1]
- RGB values will be in the range [0,1]
- RGB values will be linear, this means a gamma correction has to be applied
- The matrix $M^{-1}$ can be found here: section 24 and $\gamma$ here: section 26

XYZ to linear rgb

$$\begin{bmatrix} r \\ g \\ b \end{bmatrix} = [M]^{-1} * \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{39}$$

Gamma correction general (do this for each channel):

$$V = v^{1/\gamma} \tag{40}$$

Gamma correction for sRGB (do this for each channel):

$$V = \begin{cases} 12.92 * v & v \leq 0.0031308 \\ 1.055 * v^{1/2.4} - 0.055 & v > 0.0031308 \end{cases} \tag{41}$$

```
ColorRGB XYZToRGB(ColorXYZ XYZ, RGBColorSpace RGBSpace)
{
    ColorRGB RGB = new ColorRGB();
    double[] linearRGB = MultiplyMatrix(RGBSpace.M1, XYZ.ValueArray);

    RGB.R = GammaCorrection(RGBSpace.Name, RGBSpace.gamma, linearRGB
        [0]);
    RGB.G = GammaCorrection(RGBSpace.Name, RGBSpace.gamma, linearRGB
        [1]);
    RGB.B = GammaCorrection(RGBSpace.Name, RGBSpace.gamma, linearRGB
        [2]);

    return RGB;
}
```

With:

```
double[] MultiplayMatrix(double[,] M, double[] Color)
{
    double[] c = new double[3];
    c[0] = Color[0] * M[0, 0] + Color[1] * M[0, 1] + Color[2] * M[0,
        2];
    c[1] = Color[0] * M[1, 0] + Color[1] * M[1, 1] + Color[2] * M[1,
        2];
    c[2] = Color[0] * M[2, 0] + Color[1] * M[2, 1] + Color[2] * M[2,
        2];
    return c;
}
```

And:

```
double GammaCorrection(string SpaceName, double gamma, double value)
{
    if (SpaceName == "sRGB")
    {
        if (value <= 0.0031308) return 12.92 * value;
        else return 1.055 * Math.Pow(value, 1 / 2.4) - 0.055;
    }
    else return Math.Pow(value, 1 / gamma);
}
```

## 5.2. RGB to XYZ

- RGB values are given in the range [0,1]
- RGB values are usually gamma-corrected and have to be linearised before
- XYZ values will be in the range [0,1]
- The matrix M can be found here: section 24 and $\gamma$ here: section 26

Linearise RGB general (do this for each channel):

$$v = V^{\gamma} \tag{42}$$

Gamma correction for sRGB (do this for each channel):

$$v = \begin{cases} \dfrac{V}{12.92} & V \leq 0.04045 \\ \left(\dfrac{V + 0.055}{1.055}\right)^{2.4} & V > 0.04045 \end{cases} \tag{43}$$

linear rgb to XYZ

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = [M] * \begin{bmatrix} r \\ g \\ b \end{bmatrix} \tag{44}$$

```
ColorXYZ RGBToXYZ(ColorRGB RGB, RGBColorSpace RGBSpace)
{
    ColorXYZ XYZ = new ColorXYZ();
    double[] linearRGB = new double[3];
    linearRGB[0] = Linearise(RGBSPace.Name, RGBSpace.gamma, RGB.R);
    linearRGB[1] = Linearise(RGBSPace.Name, RGBSpace.gamma, RGB.G);
    linearRGB[2] = Linearise(RGBSPace.Name, RGBSpace.gamma, RGB.B);

    double[] tmpXYZ = MultiplyMatrix(RGBSpace.M, linearRGB);
    XZY.X = tmpXYZ[0];
    XZY.Y = tmpXYZ[1];
    XZY.Z = tmpXYZ[2];

    return XYZ;
}
```

With:

```
double[] MultiplayMatrix(double[,] M, double[] Color)
{
    double[] c = new double[3];
    c[0] = Color[0] * M[0, 0] + Color[1] * M[0, 1] + Color[2] * M[0,
        2];
    c[1] = Color[0] * M[1, 0] + Color[1] * M[1, 1] + Color[2] * M[1,
        2];
    c[2] = Color[0] * M[2, 0] + Color[1] * M[2, 1] + Color[2] * M[2,
        2];
    return c;
}
```

And:

```
double Linearise(string SpaceName, double gamma, double value)
{
    if (SpaceName == "sRGB")
    {
        if (value > 0.04045) return Math.Pow((value + 0.055) / 1.055,
            2.4);
        else return (value / 12.92);
    }
    else return Math.Pow(value, gamma);
}
```

# 6. HSV

**Range:**

- H: 0 to 360
- S: 0 to 1
- V: 0 to 1

## 6.1. RGB to HSV

- RGB values are given in the range [0,1]
- HSV.H value will be in the range [0,360] and HSV.S and HSV.V values in the range[0,1]

$$H = \begin{cases} 60 * \left( d - \dfrac{c}{max - min} \right) & max \neq min \\ 0 & max = min \end{cases} \tag{45}$$

$$S = \begin{cases} \dfrac{max - min}{max} & max \neq 0 \\ 0 & max = 0 \end{cases} \tag{46}$$

$$V = \begin{cases} max & max \neq min \\ min & max = min \end{cases} \tag{47}$$

Where:

$$max = R \vee G \vee B \tag{48}$$

$$min = R \wedge G \wedge B \tag{49}$$

And:

$$c = \begin{cases} G - B & R = min \\ B - R & G = min \\ R - G & B = min \end{cases} \tag{50}$$

$$d = \begin{cases} 3 & R = min \\ 5 & G = min \\ 1 & B = min \end{cases} \tag{51}$$

```
ColorRGB RGBToHSV(ColorRGB RGB)
{
    ColorHSV HSV = new ColorHSV();
    double max = Math.Max(RGB.R, Math.Max(RGB.G, RGB.B));
    double min = Math.Min(RGB.R, Math.Min(RGB.G, RGB.B));

    if (min == max) { HSV.V = min; HSV.H = 0; }
    else
```

```
    {
        double c;
        if (RGB.R == min) c = RGB.G - RGB.B;
        else if (RGB.B == min) c = RGB.R - RGB.G;
        else c = RGB.B - RGB.R;
        double d;
        if (RGB.R == min) d = 3;
        else if (RGB.B == min) d = 1;
        else d = 5;
        HSV.H = 60d * (d - c / (max - min));
        HSV.V = max;
    }
    if (max == 0) HSV.S = 0;
    else HSV.S = (max - min) / max;

    return HSV;
}
```

## 6.2. HSV to RGB

- HSV.H value is given in the range [0,360] and HSV.S and HSV.V values in the range[0,1]
- RGB values will be in the range [0,1]

$$R = \begin{cases} V & S = 0 \\ V & b = 6 \vee 0 \\ f & b = 1 \\ e & b = 2 \\ e & b = 3 \\ g & b = 4 \\ V & b = 5 \end{cases} \tag{52}$$

$$G = \begin{cases} V & S = 0 \\ g & b = 6 \vee 0 \\ V & b = 1 \\ V & b = 2 \\ f & b = 3 \\ e & b = 4 \\ e & b = 5 \end{cases} \tag{53}$$

$$B = \begin{cases} V & S = 0 \\ e & b = 6 \vee 0 \\ e & b = 1 \\ g & b = 2 \\ V & b = 3 \\ V & b = 4 \\ f & b = 5 \end{cases} \tag{54}$$

Where:

$$a = 6 * \frac{H}{360} \tag{55}$$

$$b = \lfloor a \rfloor \tag{56}$$

$$e = V * (1 - S) \tag{57}$$

$$f = V * (1 - S * (a - b)) \tag{58}$$

$$g = V * (1 - S * (1 - (a - b))) \tag{59}$$

```
ColorRGB HSVToRGB(ColorHSV HSV)
{
    ColorRGB RGB = new ColorRGB();
    if (HSV.S == 0)
    {
        RGB.R = HSV.V;
        RGB.G = HSV.V;
        RGB.B = HSV:V;
    }
    else
    {
        double a = (HSV.H / 360d) * 6;
        double b = Math.Floor(a);
        double e = HSV.V * (1 - HSV.S);
        double f = HSV.V * (1 - HSV.S * (a - b));
        double g = HSV.V * (1 - HSV.S * (1 - (a - b)));

        switch ((int)b)
        {
            case 6:
            case 0: RGB.R = HSV.V; RGB.G = g; RGB.B = e; break;
            case 1: RGB.R = f; RGB.G = HSV.V; RGB.B = e; break;
            case 2: RGB.R = e; RGB.G = HSV.V; RGB.B = g; break;
            case 3: RGB.R = e; RGB.G = f; RGB.B = HSV.V; break;
            case 4: RGB.R = g; RGB.G = e; RGB.B = HSV.V; break;
            default: RGB.R = HSV.V; RGB.G = e; RGB.B = f; break;
        }
    }

    return RGB;
}
```

# 7. HSL

**Range:**

- H: 0 to 360
- S: 0 to 1
- L: 0 to 1

## 7.1. RGB to HSL

- RGB values are given in the range [0,1]
- HSL.H value will be in the range [0,360] and HSL.S and HSL.L values in the range[0,1]

$$H = \begin{cases} 0 & min = max \\ 60 * \dfrac{G - B}{max - min} & R = max \\ 60 * (2 + \dfrac{B - R}{max - min}) & G = max \\ 60 * (4 + \dfrac{R - G}{max - min}) & B = max \end{cases} \tag{60}$$

$$S = \begin{cases} 0 & min = max \\ \dfrac{max - min}{max + min} & \dfrac{max + min}{2} \leq 0.5 \\ \dfrac{max - min}{2 - max - min} & \dfrac{max + min}{2} > 0.5 \end{cases} \tag{61}$$

$$L = \frac{max + min}{2} \tag{62}$$

Where:

$$max = R \vee G \vee B \tag{63}$$

$$min = R \wedge G \wedge B \tag{64}$$

```
ColorHSL RGBToHSL(ColorRGB RGB)
{
    ColorHSL HSL = new ColorHSL();
    double max = Math.Max(RGB.R, Math.Max(RGB.G, RGB.B));
    double min = Math.Min(RGB.R, Math.Min(RGB.G, RGB.B));

    if (max == min) { HSL.S = 0; HSL.H = 0; }
    else
    {
        if ((max + min) / 2d <= 0.5) { HSL.S = (max - min) / (max + min
            ); }
        else { HSL.S = (max - min) / (2 - max - min); }

```

```
13          if (RGB.R == max) { HSL.H = (RGB.G - RGB.B) / (max - min); }
14          else if (RGB.G == max) { HSL.H = 2 + (RGB.B - RGB.R) / (max -
                min); }
15          else { HSL.H = 4 + (RGB.R - RGB.G) / (max - min); }
16
17          HSL.H *= 60;
18      }
19      HSL.L = (max + min) / 2d;
20
21      return HSL;
22  }
```

## 7.2. HSL to RGB

- HSL.H value is given in the range [0,360] and HSL.S and HSL.L values in the range[0,1]
- RGB values will be in the range [0,1]

$$R = \begin{cases} L & S = 0 \\ f_{H+120} & S \neq 0 \end{cases} \tag{65}$$

$$G = \begin{cases} L & S = 0 \\ f_{H} & S \neq 0 \end{cases} \tag{66}$$

$$B = \begin{cases} L & S = 0 \\ f_{H-120} & S \neq 0 \end{cases} \tag{67}$$

Where:
(Note: bring x into the range [0,360] first)

$$f_x = \begin{cases} b + (a-b) * \dfrac{x}{60} & x < 60 \\ a & x < 180 \\ b + (a-b) * \dfrac{240 - x}{60} & x < 240 \end{cases} \tag{68}$$

And:

$$a = \begin{cases} L * (1 + S) & L < 0.5 \\ L + S - (S * L) & L \geq 0.5 \end{cases} \tag{69}$$

$$b = 2 * L - a \tag{70}$$

```
1  ColorRGB HSLToRGB(ColorHSL HSL)
2  {
3      ColorRGB RGB = new ColorRGB();
4      if (HSL.S == 0)
5      {
6          RGB.R = HSL.L;
7          RGB.G = HSL.L;
```

```
        RGB.B = HSL.L;
    }
    else
    {
        double a;
        if (HSL.L < 0.5) { a = HSL.L * (1 + HSL.S); }
        else { a = (HSL.L + HSL.S) - (HSL.S * HSL.L); }

        double b = 2 * HSL.L - a;

        RGB.R = HueToRGB(b, a, HSL.H + 120);
        RGB.G = HueToRGB(b, a, HSL.H);
        RGB.B = HueToRGB(b, a, HSL.H - 120);
    }

    return RGB;
}
```

Where:

```
double HueToRGB(double v1, double v2, double vH)
{
    if (vH > 360) { vH -= 360; }
    else if (vH < 0) { vH += 360; }

    if (vH < 60) { v1 = v1 + (v2 - v1) * vH / 60; }
    else if (vH < 180) { v1 = v2; }
    else if (vH < 240) { v1 = v1 + (v2 - v1) * (240 - vH) / 60; }

    return v1;
}
```

# 8. YCbCr

Digital YCbCr, this means values are valid from 0 to 1 **Range:**

- Y: 0 to 1
- Cb: 0 to 1
- Cr: 0 to 1

## 8.1. RGB to YCbCr

- RGB values are given in the range [0,1]
- RGB values are usually gamma-corrected and have to be linearised before
- YCbCr values will be in the range [0,1]
- The values $K_R$, $K_G$, $K_B$ and $\gamma$ can be found here: section 25

Linearise RGB general (do this for each channel):

$$v = V^{\gamma} \tag{71}$$

Gamma correction for sRGB (do this for each channel):

$$v = \begin{cases} \dfrac{V}{12.92} & V \leq 0.04045 \\ \left(\dfrac{V + 0.055}{1.055}\right)^{2.4} & V > 0.04045 \end{cases} \tag{72}$$

linear rgb to YCbCr

$$Y = K_R * r + K_G * g + K_B * b \tag{73}$$

$$C_b = \frac{b - Y}{1 - K_B} + 0.5 \tag{74}$$

$$C_r = \frac{r - Y}{1 - K_R} + 0.5 \tag{75}$$

```
ColorYCbCr RGBToYCbCr(ColorRGB RGB, YCbCrColorSpace YCbCrSpace,
    RGBColorSpace RGBSpace)
{
    ColorYCbCr YCbCr = new ColorYCbCr();
    double[] linearRGB = new double[3];
    linearRGB[0] = Linearise(RGBSPace.Name, RGBSpace.gamma, RGB.R);
    linearRGB[1] = Linearise(RGBSPace.Name, RGBSpace.gamma, RGB.G);
    linearRGB[2] = Linearise(RGBSPace.Name, RGBSpace.gamma, RGB.B);

    YCbCr.Y = YCbCrSpace.KR * linearRGB[0] + YCbCrSpace.KG * linearRGB
        [1] + YCbCrSpace.KB * linearRGB[2];
    YCbCr.Cb = (((linearRGB[2] - YCbCr.Y) / (1 - YCbCrSpace.KB)) / 2) +
        0.5;
    YCbCr.Cr = (((linearRGB[0] - YCbCr.Y) / (1 - YCbCrSpace.KR)) / 2) +
        0.5;

    return YCbCr;
}
```

Where:

```
double Linearise(string SpaceName, double gamma, double value)
{
    if (SpaceName == "sRGB")
    {
        if (value > 0.04045) return Math.Pow((value + 0.055) / 1.055,
            2.4);
        else return (value / 12.92);
    }
    else return Math.Pow(value, gamma);
}
```

## 8.2. YCbCr to RGB

- YCbCr values are given in the range [0,1]

- RGB values will be in the range [0,1]
- RGB values will be linear, this means a gamma correction has to be applied
- The values $K_R$, $K_G$, $K_B$ and $\gamma$ can be found here: section 25

YCbCr to linear rgb

$$b = -2 * C_b * (K_B - 1) + K_B + Y - 1 \tag{76}$$

$$r = -2 * C_r * (K_R - 1) + K_R + Y - 1 \tag{77}$$

$$r = \frac{(-b * K_B) - (K_R * r) + Y}{K_G} \tag{78}$$

Gamma correction general (do this for each channel):

$$V = v^{1/\gamma} \tag{79}$$

Gamma correction for sRGB (do this for each channel):

$$V = \begin{cases} 12.92 * v & v \leq 0.0031308 \\ 1.055 * v^{1/2.4} - 0.055 & v > 0.0031308 \end{cases} \tag{80}$$

```
ColorRGB YCbCrToRGB(ColorYCbCr YCbCr, YCbCrColorSpace YCbCrSpace,
    RGBColorSpace RGBSpace)
{
    ColorRGB RGB = new ColorRGB();
    double[] linearRGB = new double[3];
    linearRGB[2] = -2 * YCbCr.Cb * (YCbCrSpace.KB - 1) + YCbCrSpace.KB
        + YCbCr.Y - 1;
    linearRGB[0] = -2 * YCbCr.Cr * (YCbCrSpace.KR - 1) + YCbCrSpace.KR
        + YCbCr.Y - 1;
    linearRGB[1] = ((-linearRGB[2] * YCbCrSpace.KB) - (YCbCrSpace.KR *
        linearRGB[0]) + YCbCr.Y) / YCbCrSpace.KG;

    RGB.R = GammaCorrection(RGBSpace.Name, RGBSpace.gamma, linearRGB
        [0]);
    RGB.G = GammaCorrection(RGBSpace.Name, RGBSpace.gamma, linearRGB
        [1]);
    RGB.B = GammaCorrection(RGBSpace.Name, RGBSpace.gamma, linearRGB
        [2]);

    return RGB;
}
```

Where:

```
double GammaCorrection(string SpaceName, double gamma, double value)
{
    if (SpaceName == "sRGB")
    {
        if (value <= 0.0031308) return 12.92 * value;
```

```
        else return 1.055 * Math.Pow(value, 1 / 2.4) - 0.055;
    }
    else return Math.Pow(value, 1 / gamma);
}
```

# 9. LCHab

**Range:**

- L: 0 to 100
- C: not defined
- H: 0 to 360

## 9.1. Lab to LCHab

- Lab.L value is given in the range [0,100]
- LCH.L value will be in the range [0,100], LCH.H in the range [0,360]
- watch out for a = 0 when calculating H

$$L = L \tag{81}$$

$$C = \sqrt{a^2 + b^2} \tag{82}$$

$$H = \arctan \frac{b}{a} \tag{83}$$

```
ColorLCH LabToLCHab(ColorLab Lab)
{
    ColorLCH LCH = new ColorLCH();
    LCH.L = Lab.L;
    LCH.C = Math.Sqrt(Math.Pow(Lab.a, 2) + Math.Pow(Lab.b, 2));
    LCH.H = Math.Atan2(Lab.b, Lab.a) * 180 / Math.PI;

    return LCH;
}
```

## 9.2. LCHab to Lab

- LCH.L value is given in the range [0,100], LCH.H in the range [0,360]
- Lab.L value will be in the range [0,100]

$$L = L \tag{84}$$

$$a = C * \cos H \tag{85}$$

$$b = C * \sin H \tag{86}$$

```
ColorLab LCHabToLab(ColorLCH LCH)
{
    ColorLab Lab = new ColorLab();
    Lab.L = LCH.L;
    Lab.a = LCH.C * Math.Cos(LCH.H * Math.PI / 180d);
    Lab.b = LCH.C * Math.Sin(LCH.H * Math.PI / 180d);

    return Lab;
}
```

# 10. LCHuv

**Range:**

- L: 0 to 100
- C: not defined
- H: 0 to 360

## 10.1. Luv to LCHuv

- Luv.L value is given in the range [0,100]
- LCH.L value will be in the range [0,100], LCH.H in the range [0,360]
- watch out for a = 0 when calculating H

$$L = L \tag{87}$$

$$C = \sqrt{u^2 + v^2} \tag{88}$$

$$H = \arctan \frac{u}{v} \tag{89}$$

```
ColorLCH LCHuvToLuv(ColorLuv Luv)
{
    ColorLCH LCH = new ColorLCH();
    LCH.L = Luv.L;
    LCH.C = Math.Sqrt(Math.Pow(Luv.u, 2) + Math.Pow(Luv.v, 2));
    LCH.H = Math.Atan2(Luv.v, Luv.u) * 180 / Math.PI;

    return LCH;
}
```

## 10.2. LCHuv to Luv

- LCH.L value is given in the range [0,100], LCH.H in the range [0,360]
- Luv.L value will be in the range [0,100]

$$L = L \tag{90}$$

$$u = C * \cos H \tag{91}$$

$$v = C * \sin H \tag{92}$$

```
ColorLuv LCHuvToLuv(ColorLCH LCH)
{
    ColorLuv Luv = new ColorLuv();
    Luv.L = LCH.L;
    Luv.u = LCH.C * Math.Cos(LCH.H * Math.PI / 180d);
    Luv.v = LCH.C * Math.Sin(LCH.H * Math.PI / 180d);

    return Luv;
}
```

# 11. DIN99

**Range:**

- L: 0 to 100
- C: not defined
- H: 0 to 360

## 11.1. Lab to DIN99

- Lab.L value is given in the range [0,100]
- LCH.L value will be in the range [0,100], LCH.H in the range [0,360]

$$L_{99} = 105.51 * \ln(1 + 0.0158 * L) \tag{93}$$

$$C = \frac{\ln(1 + 0.045 * \sqrt{e^2 + f^2})}{0.045} \tag{94}$$

$$H = (\frac{180}{\pi} * \arctan\frac{f}{e}) + 0.045 \tag{95}$$

Where:

$$e = a * \cos 16 + b * \sin 16 \tag{96}$$

$$f = 0.7 * (-a * \sin 16 + b * \cos 16) \tag{97}$$

```
ColorLCH LabToLCH99(ColorLab Lab)
{
    ColorLCH LCH = new ColorLCH();
    LCH.L = 105.51 * Math.Log(1 + 0.0158 * Lab.L);
    double e = Lab.a * Math.Cos(16) + Lab.b * Math.Sin(16);
    double f = 0.7 * (-Lab.a * Math.Sin(16) + Lab.b * Math.Cos(16));
    LCH.C = Math.Log(1 + 0.045 * Math.Sqrt(Math.Pow(e,2) + Math.Pow(f
        ,2))) / 0.045;
    LCH.H = (Math.Atan2(f, e) * 180/Math.PI);

    return LCH;
}
```

## 11.2. DIN99 to Lab

- LCH.L value is given in the range [0,100], LCH.H in the range [0,360]
- Lab.L value will be in the range [0,100]

$$L = \frac{\exp\left(\frac{L_{99}}{105.51}\right) - 1}{0.0158} \tag{98}$$

$$a = e * \cos 16 - \frac{f}{0.7} * \sin 16 \tag{99}$$

$$b = e * \sin 16 + \frac{f}{0.7} * \cos 16 \tag{100}$$

Where:

$$d = \frac{\exp(0.045 * C) - 1}{0.045} \tag{101}$$

$$g = H * \frac{\pi}{180} \tag{102}$$

$$e = d * \cos g \tag{103}$$

$$f = d * \sin g \tag{104}$$

```
ColorLab LCH99ToLab(ColorLCH LCH)
{
    ColorLab Lab = new ColorLab();
    double d = (Math.Exp(0.045 * LCH.C) - 1) / 0.045;
    double g = LCH.H * Math.PI/180d;
    double e = d * Math.Cos(g);
    double f = d * Math.Sin(g);
    Lab.a = e * Math.Cos(16) - (f / 0.7) * Math.Sin(16);
    Lab.b = e * Math.Sin(16) + (f / 0.7) * Math.Cos(16);
    Lab.L = (Math.Exp(LCH.L / 105.51) - 1) / 0.0158;

    return Lab;
}
```

# 12. DIN99b

**Range:**

- L: 0 to 100
- C: not defined
- H: 0 to 360

## 12.1. Lab to DIN99b

- Lab.L value is given in the range [0,100]
- LCH.L value will be in the range [0,100], LCH.H in the range [0,360]

$$L_{99b} = 303.671 * \ln(1 + 0.0039 * L) \tag{105}$$

$$C = \ln(1 + 0.075 * \sqrt{e^2 + f^2}) * 23 \tag{106}$$

$$H = (\frac{180}{\pi} * \arctan\frac{f}{e}) + 26 \tag{107}$$

Where:

$$e = a * \cos 26 + b * \sin 26 \tag{108}$$

$$f = 0.83 * (-a * \sin 26 + b * \cos 26) \tag{109}$$

```
ColorLCH LabToLCH99b(ColorLab Lab)
{
    ColorLCH LCH = new ColorLCH();
    LCH.L = 303.671 * Math.Log(1 + 0.0039 * Lab.L);
    double e = Lab.a * Math.Cos(26) + Lab.b * Math.Sin(26);
    double f = 0.83 * (-Lab.a * Math.Sin(26) + Lab.b * Math.Cos(26));
    LCH.C = Math.Log(1 + 0.075 * Math.Sqrt(Math.Pow(e,2) + Math.Pow(f
        ,2))) * 23;
    LCH.H = (Math.Atan2(f, e) * 180 / Math.PI) + 26;

    return LCH;
}
```

## 12.2. DIN99b to Lab

- LCH.L value is given in the range [0,100], LCH.H in the range [0,360]
- Lab.L value will be in the range [0,100]

$$L = \frac{\exp\left(\frac{L_{99b}}{303.671}\right) - 1}{0.0039} \tag{110}$$

$$a = e * \cos 26 - \frac{f}{0.83} * \sin 26 \tag{111}$$

$$b = e * \sin 26 + \frac{f}{0.83} * \cos 26 \tag{112}$$

Where:

$$d = \frac{\exp(\frac{C}{23}) - 1}{0.075} \tag{113}$$

$$g = (H - 26) * \frac{\pi}{180} \tag{114}$$

$$e = d * \cos g \tag{115}$$

$$f = d * \sin g \tag{116}$$

```
ColorLab LCH99bToLab(ColorLCH LCH)
{
    ColorLab Lab = new ColorLab();
    double d = (Math.Exp(LCH.C) / 23d - 1) / 0.075;
    double g = (LCH.H - 26) * Math.PI/180d;
    double e = d * Math.Cos(g);
    double f = d * Math.Sin(g);
    Lab.a = e * Math.Cos(26) - (f / 0.83) * Math.Sin(26);
    Lab.b = e * Math.Sin(26) + (f / 0.83) * Math.Cos(26);
    Lab.L = (Math.Exp(LCH.L / 303.671) - 1) / 0.0039;

    return Lab;
}
```

# 13. DIN99c

**Range:**

- L: 0 to 100
- C: not defined
- H: 0 to 360

## 13.1. Lab to DIN99c

- Lab.L value is given in the range [0,100]
- LCH.L value will be in the range [0,100], LCH.H in the range [0,360]
- Before converting, XYZ.X has to be adjusted: $X' = 1.1 * X - 0.1 * Z$ This means, from Lab, you'll have to convert to XYZ, do adjustment, convert back to Lab and then convert to DIN99c.

$$L_{99c} = 317.651 * \ln(1 + 0.0037 * L) \tag{117}$$

$$C = \ln(1 + 0.066 * \sqrt{e^2 + f^2}) * 23 \tag{118}$$

$$H = \frac{180}{\pi} * \arctan\frac{f}{e} \tag{119}$$

Where:

$$e = a \tag{120}$$

$$f = 0.94 * b \tag{121}$$

```
ColorLCH LabToLCH99c(ColorLab Lab)
{
    ColorLCH LCH = new ColorLCH();
    LCH.L = 317.651 * Math.Log(1 + 0.0037 * Lab.L);
    double e = Lab.a;
    double f = 0.94 * Lab.b;
    LCH.C = Math.Log(1 + 0.066 * Math.Sqrt(Math.Pow(e, 2) + Math.Pow(f,
        2))) * 23;
    LCH.H = Math.Atan2(f, e) * 180 / Math.PI;

    return LCH;
}
```

## 13.2. DIN99c to Lab

- LCH.L value is given in the range [0,100], LCH.H in the range [0,360]
- Lab.L value will be in the range [0,100]
- After converting to Lab, you'll have to convert to XYZ and adjust XYZ.X like this: $X' = \frac{X + 0.12 * Z}{1.12}$ After that you may convert back to Lab if you need it.

$$L = \frac{\exp\left(\frac{L_{99c}}{317.651}\right) - 1}{0.0037} \tag{122}$$

$$a = d * \sin g \tag{123}$$

$$b = \frac{f}{0.94} \tag{124}$$

Where:

$$d = \frac{\exp(\frac{C}{23}) - 1}{0.066} \tag{125}$$

$$g = H * \frac{\pi}{180} \tag{126}$$

$$f = d * \sin g \tag{127}$$

```
ColorLab LCH99cToLab(ColorLCH LCH)
{
    ColorLab Lab = new ColorLab();
    double d = (Math.Exp(LCH.C / 23d) - 1) / 0.066;
    double g = LCH.H * Math.PI / 180d;
    double f = d * Math.Sin(g);
    Lab.a = d * Math.Cos(g);
    Lab.b = f / 0.94;
    Lab.L = (Math.Exp(LCH.L / 317.651) - 1) / 0.0037;

    return Lab;
}
```

# 14. DIN99d

**Range:**

- L: 0 to 100
- C: not defined
- H: 0 to 360

## 14.1. Lab to DIN99d

- Lab.L value is given in the range [0,100]
- LCH.L value will be in the range [0,100], LCH.H in the range [0,360]
- Before converting, XYZ.X has to be adjusted: $X' = 1.12 * X - 0.12 * Z$ This means, from Lab, you'll have to convert to XYZ, do adjustment, convert back to Lab and then convert to DIN99d.

$$L_{99d} = 325.221 * \ln(1 + 0.0036 * L) \tag{128}$$

$$C = \ln(1 + 0.06 * \sqrt{e^2 + f^2}) * 22.5 \tag{129}$$

$$H = (\frac{180}{\pi} * \arctan\frac{f}{e}) + 16 \tag{130}$$

Where:

$$e = a * \cos 50 + b * \sin 50 \tag{131}$$

$$f = 1.14 * (-a * \sin 50 + b * \cos 50) \tag{132}$$

```
ColorLCH LabToLCH99d(ColorLab Lab)
{
    ColorLCH LCH = new ColorLCH();
    LCH.L = 325.221 * Math.Log(1 + 0.0036 * Lab.L);
    double e = Lab.a * Math.Cos(50) + Lab.b * Math.Sin(50);
    double f = 1.14 * (-Lab.a * Math.Sin(50) + Lab.b * Math.Cos(50));
    LCH.C = Math.Log(1 + 0.06 * Math.Sqrt(Math.Pow(e, 2) + Math.Pow(f,
        2))) * 22.5;
    LCH.H = (Math.Atan2(f, e) * 180 / Math.PI) + 16;

    return LCH;
}
```

## 14.2. DIN99d to Lab

- LCH.L value is given in the range [0,100], LCH.H in the range [0,360]
- Lab.L value will be in the range [0,100]
- After converting to Lab, you'll have to convert to XYZ and adjust XYZ.X like this: $X' = \frac{X + 0.1 * Z}{1.1}$ After that you may convert back to Lab if you need it.

$$L = \frac{\exp\left(\frac{L_{99d}}{325.22}\right) - 1}{0.0036} \tag{133}$$

$$a = e * \cos 50 - \frac{f}{1.14} * \sin 50 \tag{134}$$

$$b = e * \sin 50 + \frac{f}{1.14} * \cos 50 \tag{135}$$

Where:

$$d = \frac{\exp(\frac{C}{22.5}) - 1}{0.06} \tag{136}$$

$$g = (H - 16) * \frac{\pi}{180} \tag{137}$$

$$e = d * \cos g \tag{138}$$

$$f = d * \sin g \tag{139}$$

```
ColorLab LCH99dToLab(ColorLCH LCH)
{
    ColorLab Lab = new ColorLab();
    double d = (Math.Exp(LCH.C / 22.5) - 1) / 0.06;
    double g = (LCH.H - 16) * Math.PI/180d;
    double e = d * Math.Cos(g);
    double f = d * Math.Sin(g);
    Lab.a = e * Math.Cos(50) - (f / 1.14) * Math.Sin(50);
    Lab.b = e * Math.Sin(50) + (f / 1.14) * Math.Cos(50);
    Lab.L = (Math.Exp(LCH.L / 325.22) - 1) / 0.0036;

    return Lab
}
```

# 15. DEF

**Range:**

- D: not defined
- E: not defined
- F: not defined

## 15.1. XYZ to DEF

- XYZ values are given in the range [0,1]

$$\begin{bmatrix} D \\ E \\ F \end{bmatrix} = [M] * \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{140}$$

Where:

$$[M] = \begin{bmatrix} 0.2053 & 0.7125 & 0.4670 \\ 1.8537 & -1.2797 & -0.4429 \\ -0.3655 & 1.0120 & -0.6104 \end{bmatrix} \tag{141}$$

```
ColorDEF XYZToDEF(ColorXYZ XYZ)
{
    ColorDEF DEF = new ColorDEF();
    double[] tmp = MultiplyMatrix(M, XYZ.ValueArray);
    DEF.D = tmp[0];
    DEF.E = tmp[1];
    DEF.F = tmp[2];

    return DEF;
}
```

With:

```
double[] MultiplayMatrix(double[,] M, double[] Color)
{
    double[] c = new double[3];
    c[0] = Color[0] * M[0, 0] + Color[1] * M[0, 1] + Color[2] * M[0,
        2];
    c[1] = Color[0] * M[1, 0] + Color[1] * M[1, 1] + Color[2] * M[1,
        2];
    c[2] = Color[0] * M[2, 0] + Color[1] * M[2, 1] + Color[2] * M[2,
        2];
    return c;
}
```

## 15.2. DEF to XYZ

- XYZ values will be in the range [0,1]

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} M \end{bmatrix}^{-1} * \begin{bmatrix} D \\ E \\ F \end{bmatrix} \tag{142}$$

Where:

$$\begin{bmatrix} M \end{bmatrix}^{-1} = \begin{bmatrix} 0.671203 & 0.495489 & 0.153997 \\ 0.706165 & 0.0247732 & 0.522292 \\ 0.768864 & -0.255621 & -0.864558 \end{bmatrix} \tag{143}$$

```
ColorXYZ DEFToXYZ(ColorDEF DEF)
{
    ColorXYZ XYZ = new ColorXYZ();
    double[] tmp = MultiplyMatrix(M_1, DEF.ValueArray);
    XYZ.X = tmp[0];
    XYZ.Y = tmp[1];
    XYZ.Z = tmp[2];

    return XYZ;
}
```

With:

```
double[] MultiplayMatrix(double[,] M, double[] Color)
{
    double[] c = new double[3];
    c[0] = Color[0] * M[0, 0] + Color[1] * M[0, 1] + Color[2] * M[0,
        2];
    c[1] = Color[0] * M[1, 0] + Color[1] * M[1, 1] + Color[2] * M[1,
        2];
    c[2] = Color[0] * M[2, 0] + Color[1] * M[2, 1] + Color[2] * M[2,
        2];
    return c;
}
```

# 16. Bef

**Range:**

- B: not defined
- e: not defined
- f: not defined

## 16.1. DEF to Bef

- Bef values will be positive values

$$B = \sqrt{D^2 + E^2 + F^2} \tag{144}$$

$$e = E/B \tag{145}$$

$$f = F/B \tag{146}$$

```
ColorBef DEFToBef(ColorDEF DEF)
{
    ColorBef Bef = new ColorBef();
    Bef.B = Math.Sqrt(Math.Pow(DEF.D, 2) + Math.Pow(DEF.E, 2) + Math.
        Pow(DEF.F, 2));
    Bef.e = DEF.E / Bef.B;
    Bef.f = DEF.F / Bef.B;

    return Bef;
}
```

## 16.2. Bef to DEF

- Bef values are given as positive values

$$E = e * B \tag{147}$$

$$F = f * B \tag{148}$$

$$D = \sqrt{B^2 - E^2 - F^2} \tag{149}$$

```
ColorDEF BefToDEF(ColorBef Bef)
{
    ColorDEF DEF = new ColorDEF();
    DEF.E = Bef.e * Bef.B;
    DEF.F = Bef.f * Bef.B;
    DEF.D = Math.Sqrt(Math.Pow(Bef.B, 2) - Math.Pow(DEF.E, 2) - Math.
        Pow(DEF.F, 2));

    return DEF;
}
```

# 17. BCH

**Range:**

- B: not defined
- C: not defined
- H: 0 to 360

## 17.1. DEF to BCH

- BCH.B value will be a positive value, BCH.H in the range [0,360]

$$B = \sqrt{D^2 + E^2 + F^2} \tag{150}$$

$$C = \begin{cases} \arcsin\left(\dfrac{\sqrt{E^2+F^2}}{B} * -1\right) & F < 0 \\ \arcsin\left(\dfrac{\sqrt{E^2+F^2}}{B}\right) & F \geq 0 \end{cases} \tag{151}$$

$$H = \begin{cases} \arccos\left(\dfrac{E}{\sqrt{E^2+F^2}} * -1\right) & F < 0 \\ \arccos\left(\dfrac{E}{\sqrt{E^2+F^2}}\right) & F \geq 0 \end{cases} \tag{152}$$

```
ColorBCH DEFToBCH(ColorDEF DEF)
{
    ColorBCH BCH = new ColorBCH();
    BCH.B = Math.Sqrt(Math.Pow(DEF.D, 2) + Math.Pow(DEFE, 2) + Math.Pow
        (DEF.F, 2));
    BCH.C = Math.Asin((Math.Sqrt(Math.Pow(DEF.E, 2) + Math.Pow(DEF.F,
        2)) / BCH.B) * Sign(DEF.F));
    BCH.H = Math.Acos(DEF.E / Math.Sqrt(Math.Pow(DEF.E, 2) + Math.Pow(
        DEF.F, 2)) * Sign(DEF.F));

    return BCH;
}
```

Where:

```
private static int Sign(double d)
{
    if (d < 0) return -1;
    else return 1;
}
```

## 17.2. BCH to DEF

- BCH.B value is given as positive value, BCH.H in the range [0,360]

$$D = B * \sin\frac{\pi}{2} - C \tag{153}$$

$$E = B * \sin C * \cos H \tag{154}$$

$$F = B * \sin C * \sin H \tag{155}$$

```
ColorDEF BCHToDEF(ColorBCH BCH)
{
    ColorDEF DEF = new ColorDEF();
    double tmp = BCH.B * Math.Sin(BCH.C);
    DEF.D = BCH.B * Math.Sin((Math.PI / 2d) - BCH.C);
    DEF.E = tmp * Math.Cos(BCH.H);
    DEF.F = tmp * Math.Sin(BCH.H);

    return DEF;
}
```

# Part IV.
# Color Difference

## 18. DIN99

Given two DIN99 LCH color like described in Part III

### 18.1. Delta E

$$\Delta E_{99} = \sqrt{\Delta L^2 + \Delta a_{99}^2 + \Delta b_{99}^2} \tag{156}$$

Where:

$$a_{99} = C * \cos(H * \frac{\pi}{180}) \tag{157}$$

$$b_{99} = C * \sin(H * \frac{\pi}{180}) \tag{158}$$

```
double GetDeltaE_DIN99(ColorLCH LCH1, ColorLCH LCH2)
{
    double a99_1 = LCH_1.C * Math.Cos(Color1.H * Pi180);
    double b99_1 = LCH_1.C * Math.Sin(Color1.H * Pi180);
    double a99_2 = LCH_2.C * Math.Cos(Color2.H * Pi180);
    double b99_2 = LCH_2.C * Math.Sin(Color2.H * Pi180);
    return Math.Sqrt(Math.Pow(LCH_1.L - LCH_2.L, 2) + Math.Pow(a99_1 -
        a99_2, 2) + Math.Pow(a99_2 - b99_2, 2));
}
```

### 18.2. Delta C

$$\Delta C = C_1 - C_2 \tag{159}$$

```
double GetDeltaC_DIN99(ColorLCH LCH1, ColorLCH LCH2)
{
    return LCH_1.C - LCH_2.C;
}
```

### 18.3. Delta H

$$\Delta H_{99} = \sqrt{0.5 * (C_2 * C_1 + a_{99_1} * a_{99_2} + b_{99_1} * b_{99_2})} \tag{160}$$

Where:

$$a_{99} = C * \cos(H * \frac{\pi}{180}) \tag{161}$$

$$b_{99} = C * \sin(H * \frac{\pi}{180}) \tag{162}$$

```
double GetDeltaH_DIN99(ColorLCH LCH1, ColorLCH LCH2)
{
    double a99_1 = LCH_1.C * Math.Cos(Color1.H * Pi180);
    double b99_1 = LCH_1.C * Math.Sin(Color1.H * Pi180);
    double a99_2 = LCH_2.C * Math.Cos(Color2.H * Pi180);
    double b99_2 = LCH_2.C * Math.Sin(Color2.H * Pi180);

    double t = Math.Sqrt(0.5 * (LCH_2.C * LCH_1.C + a99_2 * a99_1 +
        b99_2 * b99_1));

    double H;
    if (t == 0) H = 0;
    else H = (a99_1 * b99_2 - a99_2 * b99_1) / t;

    return H;
}
```

# 19. CIE 76

Given two Lab colors like described in Part III

## 19.1. Delta E

$$\Delta E = \sqrt{\Delta L^2 + \Delta a^2 + \Delta b^2} \tag{163}$$

```
double GetDeltaE_CIE76(ColorLab Lab1, ColorLab Lab2)
{
    return Math.Sqrt(Math.Pow(Lab2.L - Lab1.L, 2) + Math.Pow(Lab2.a -
        Lab1.a, 2) + Math.Pow(Lab2.b - Lab1.b, 2));
}
```

# 20. CIE 94

Given two Lab colors like described in Part III

## 20.1. Delta E

$$\Delta E = \sqrt{\frac{\Delta L}{S_L}^2 + \left(\frac{\Delta C}{1 + K_1 * C_1}\right)^2 + \frac{\Delta H}{(1 + K_2 * C_1)^2}} \tag{164}$$

Where:

$$C = \sqrt{a^2 + b^2} \tag{165}$$

And:

$$S_L = \begin{cases} 2 & Textiles \\ 1 & GraphicArts \end{cases} \tag{166}$$

$$K_1 = \begin{cases} 0.048 & Textiles \\ 0.045 & GraphicArts \end{cases} \tag{167}$$

$$K_2 = \begin{cases} 0.014 & Textiles \\ 0.015 & GraphicArts \end{cases} \tag{168}$$

```
double GetDeltaE_CIE94(ColorLab Lab1, ColorLab Lab2)
{
    double C1 = Math.Sqrt(Math.Pow(Lab1.a, 2) + Math.Pow(Lab1.b, 2));
    double C2 = Math.Sqrt(Math.Pow(Lab2.a, 2) + Math.Pow(Lab2.b, 2));
    H = Math.Pow(Lab1.a - Lab2.a, 2) + Math.Pow(Lab1.b - Lab2.b, 2) -
       Math.Pow(C1 - C2, 2);

    return Math.Sqrt(Math.Pow((Lab1.L - Lab2.L) / SL, 2) + Math.Pow((C1
        - C2) / (1 + K1 * C1), 2) + H / Math.Pow((1 + K2 * C1), 2));
}
```

## 20.2. Delta C

$$\Delta C = C_1 - C_2 \tag{169}$$

Where:

$$C = \sqrt{a^2 + b^2} \tag{170}$$

```
double GetDeltaC_CIE94(ColorLab Lab1, ColorLab Lab2)
{
    double C1 = Math.Sqrt(Math.Pow(Lab1.a, 2) + Math.Pow(Lab1.b, 2));
    double C2 = Math.Sqrt(Math.Pow(Lab2.a, 2) + Math.Pow(Lab2.b, 2));
    return C1 - C2;
}
```

## 20.3. Delta H

$$\Delta H = \sqrt{(a_1 - a_2)^2 + (b_1 - b_2)^2 - (C_1 - C_2)^2} \tag{171}$$

Where:

$$C = \sqrt{a^2 + b^2} \tag{172}$$

```
double GetDeltaH_CIE94(ColorLab Lab1, ColorLab Lab2)
{
    double C1 = Math.Sqrt(Math.Pow(Lab1.a, 2) + Math.Pow(Lab1.b, 2));
    double C2 = Math.Sqrt(Math.Pow(Lab2.a, 2) + Math.Pow(Lab2.b, 2));
    double H = Math.Pow(Lab1.a - Lab2.a, 2) + Math.Pow(Lab1.b - Lab2.b,
        2) - Math.Pow(C1 - C2, 2);
    if (H <= 0) H = 0;
    else H = Math.Sqrt(H);

    return H;
}
```

# 21. CIE DE 2000

Given two Lab colors like described in Part III

## 21.1. Delta E

$$\Delta E = \sqrt{\left(\frac{\Delta L}{k_L * S_L}\right)^2 + \left(\frac{\Delta C_{ab}}{k_C * S_C}\right)^2 + \left(\frac{\Delta H_{ab}}{k_H * S_H}\right)^2 + R_T * \frac{\Delta C'}{k_C * S_C} * \frac{\Delta H'}{k_H * S_H}} \tag{173}$$

Where:

$$\Delta L' = L_2 - L_1 \tag{174}$$

$$\bar{L} = \frac{L_1 + L_2}{2} \tag{175}$$

$$\bar{C} = \frac{C1 + C2}{2} \tag{176}$$

$$a'_1 = a_1 + \frac{a_1}{2} * \left(1 - \sqrt{\frac{\bar{C}^7}{\bar{C}^7 + 25^7}}\right) \tag{177}$$

$$a'_2 = a_2 + \frac{a_2}{2} * \left(1 - \sqrt{\frac{\bar{C}^7}{\bar{C}^7 + 25^7}}\right) \tag{178}$$

$$\bar{C}' = \frac{C'_1 + C'_2}{2} \tag{179}$$

$$\Delta C' = C'_2 - C'_1 \tag{180}$$

$$C'_1 = \sqrt{a_1'^2 + b_1^2} \tag{181}$$

$$C'_2 = \sqrt{a_2'^2 + b_2^2} \tag{182}$$

$$h'_1 = \arctan\frac{b_1}{a'_1} \tag{183}$$

$$h'_2 = \arctan\frac{b_2}{a'_2} \tag{184}$$

$$\Delta h' = \begin{cases} h'_2 - h'_1 & |h'_1 - h'_2| \leq \pi \\ h'_2 - h'_1 + 2 * \pi & |h'_1 - h'_2| > \pi \wedge h'_2 \leq h'_1 \\ h'_2 - h'_1 - 2 * \pi & |h'_1 - h'_2| > \pi \wedge h'_2 > h'_1 \end{cases} \tag{185}$$

$$\Delta H' = 2 * \sqrt{C'_1 * C'_2} * \sin\left(\frac{\Delta h'}{2}\right) \tag{186}$$

$$\bar{H}' = \begin{cases} \dfrac{h'_1 + h'_2 + 2 * \pi}{2} & |h'_1 - h'_2| > \pi \\ \dfrac{h'_1 + h'_2}{2} & |h'_1 - h'_2| \leq \pi \end{cases} \tag{187}$$

$$T = 1 - 0.17 * \cos(\bar{H}' - 0.523599) + 0.24 * \cos(2 * \bar{H}') + 0.32 * \cos(3 * \bar{H}' + 0.10472) - 0.2 * \cos(4 * \bar{H}' - 1.099557) \tag{188}$$

$$S_L = 1 + \frac{0.015 * (\bar{L} - 50)^2}{\sqrt{20 + (\bar{L} - 50)^2}} \tag{189}$$

$$S_C = 1 + 0.045 * \bar{C}' \tag{190}$$

$$S_H = 1 + 0.015 * \bar{C}' * T \tag{191}$$

$$R_T = 2 * \sqrt{\frac{\bar{C}'^7}{\bar{C}'^7 + 25^7}} * \sin\left[1.0471976 * \exp\left(-\left[\frac{\bar{H}' - 4.799655443}{0.4363323}\right]^2\right)\right] \tag{192}$$

```
double GetDeltaE_CIEDE2000(ColorLab Lab1, ColorLab Lab2)
{
    double Pi2 = Math.PI * 2;
    double L_ = (Lab1.L + Lab2.L) / 2d;
    double C1 = Math.Sqrt(Math.Pow(Lab1.a, 2) + Math.Pow(Lab1.b, 2));
    double C2 = Math.Sqrt(Math.Pow(Lab2.a, 2) + Math.Pow(Lab2.b, 2));
    double C_ = (C1 + C2) / 2d;
    double G = (1 - Math.Sqrt((Math.Pow(C_, 7)) / (Math.Pow(C_, 7) +
       6103515625))) / 2d;
    double a1_ = Lab1.a * (1 + G);
    double a2_ = Lab2.a * (1 + G);
    double C1_ = Math.Sqrt(Math.Pow(a1_, 2) + Math.Pow(Lab1.b, 2));
    double C2_ = Math.Sqrt(Math.Pow(a2_, 2) + Math.Pow(Lab2.b, 2));
    double C__ = (C1_ + C2_) / 2d;
    double h1_ = Math.Atan2(Lab1.b, a1_);
    h1_ = (h1_ < 0) ? h1_ + Pi2 : (h1_ >= Pi2) ? h1_ - Pi2 : h1_;
    double h2_ = Math.Atan2(Lab2.b, a2_);
    h2_ = (h2_ < 0) ? h2_ + Pi2 : (h2_ >= Pi2) ? h2_ - Pi2 : h2_;
    double H__ = (Math.Abs(h1_ - h2_) > Math.PI) ? (h1_ + h2_ + Pi2) /
       2d : (h1_ + h2_) / 2d;
    double T = 1 - 0.17 * Math.Cos(H__ - 0.5236) + 0.24 * Math.Cos(2 *
       H__) + 0.32 * Math.Cos(3 * H__ + 0.10472) - 0.2 * Math.Cos(4 *
       H__ - 1.0995574);
```

```
20      double dh_ = h2_ - h1_;
21      dh_ = (Math.Abs(dh_) > Math.PI && h2_ <= h1_) ? dh_ + Pi2 : (Math.
            Abs(dh_) > Math.PI && h2_ > h1_) ? dh_ - Pi2 : dh_;
22      double dH_ = 2 * Math.Sqrt(C1_ * C2_) * Math.Sin(dh_ / 2d);
23      double SL = 1 + ((0.015 * Math.Pow(L_ - 50, 2)) / (Math.Sqrt(20 +
            Math.Pow(L_ - 50, 2))));
24      double SC = 1 + 0.045 * C__;
25      double SH = 1 + 0.015 * C__ * T;
26      double dO = 1.0471976 * Math.Exp(-Math.Pow((var13 - 4.799655) /
            0.436332313, 2));
27      double RC = 2 * Math.Sqrt(Math.Pow(C__, 7) / (Math.Pow(C__, 7) +
            6103515625));
28      double RT = -RC * Math.Sin(2 * dO);
29
30      return Math.Sqrt(Math.Pow((Lab2.L - Lab1.L) / SL, 2) + Math.Pow((
            C2_ - C1_) / SC, 2) + Math.Pow(dH_ / SH, 2) + RT * ((C2_ - C1_)
            / SC) * ((dH_) / SH));
31  }
```

# 22. CMC

Given two Lab colors like described in Part III Made to use with D65

## 22.1. Delta E

$$\Delta E = \sqrt{\left(\frac{L_2 - L_1}{l * S_L}\right)^2 + \left(\frac{C_2 - C_1}{c * S_C}\right)^2 + \left(\frac{\Delta H_a b}{S_H}\right)^2} \tag{193}$$

$$S_L = \begin{cases} 0.511 & L_1 < 16 \\ \dfrac{0.040975 * L_1}{1 + 0.01765 * L_1} & L_1 \geq 16 \end{cases} \tag{194}$$

$$S_C = \frac{0.0638 * C_1}{1 + 0.0131 * C_1} + 0.638 \tag{195}$$

$$S_H = S_C * (F * T + 1 - F) \tag{196}$$

$$F = \sqrt{\frac{C_1^4}{C_1^4 + 1900}} \tag{197}$$

$$T = \begin{cases} 0.56 + |0.2 * \cos(h_1 + 168\,^\circ)| & 164^\circ \leq h_1 \leq 345^\circ \\ 0.36 + |0.4 * \cos(h_1 + 35^\circ)| & \text{otherwise} \end{cases} \tag{198}$$

$$l : c = \begin{cases} 2:1 & \text{acceptability} \\ 1:1 & \text{perceptibility} \end{cases} \tag{199}$$

```
double GetDeltaE_CMC(ColorLab Lab1, ColorLab Lab2)
{
    double Pi2 = Math.PI * 2;
    double C1 = Math.Sqrt(Math.Pow(Lab1.a, 2) + Math.Pow(Lab1.b, 2));
    double C2 = Math.Sqrt(Math.Pow(Lab2.a, 2) + Math.Pow(Lab2.b, 2));
    double H = Math.Pow(Lab1.a - Lab2.a, 2) + Math.Pow(Lab1.b - Lab2.b,
        2) - Math.Pow(C1 - C2, 2);
    if (var1 < 0) H = 0;
    else H = Math.Sqrt(H);
    double SL;
    if (Lab1.L < 16) SL = 0.511;
    else SL = (Lab1.L * 0.040975) / (1 + Lab1.L * 0.01765);
    double SC = ((0.0638 * C1) / (1 + 0.0131 * C1)) + 0.638;
    h1 = Math.Atan2(Lab1.b, Lab1.a);
    if (h1 < 0) h1 = h1 + Pi2;
    else if (h1 >= Pi2) h1 = h1 - Pi2;
    doube T;
    if (h1 <= 6.021386 && h1 >= 2.86234) T = 0.56 + Math.Abs(0.2 * Math
        .Cos(h1 + 2.932153));
    else T = 0.36 + Math.Abs(0.4 * Math.Cos(h1 + 0.61086524));
    F = Math.Sqrt(Math.Pow(C1, 4) / (Math.Pow(C1, 4) + 1900));
    SH = SC * (F * T + 1 - F);

    return Math.Sqrt(Math.Pow((Lab1.L - Lab2.L) / (l * SL), 2) + Math.
        Pow((C1 - C2) / (c * SC), 2) + Math.Pow(H / SH, 2));
}
```

## 22.2. Delta C

$$\Delta C = C_1 - C_2 \tag{200}$$

$$C_1 = \sqrt{a_1^2 + b_1^2} \tag{201}$$

$$C_2 = \sqrt{a_2^2 + b_2^2} \tag{202}$$

```
double GetDeltaC_CMC(ColorLab Lab1, ColorLab Lab2)
{
    double C1 = Math.Sqrt(Math.Pow(Lab1.a, 2) + Math.Pow(Lab1.b, 2));
    double C2 = Math.Sqrt(Math.Pow(Lab2.a, 2) + Math.Pow(Lab2.b, 2));
    return C1 - C2;
}
```

## 22.3. Delta H

$$\Delta H = \sqrt{(a_1 - a_2)^2 + (b_1 - b_2)^2 - (C_1 - C_2)^2} \tag{203}$$

$$C_1 = \sqrt{a_1^2 + b_1^2} \tag{204}$$

$$C_2 = \sqrt{a_2^2 + b_2^2} \tag{205}$$

```
double GetDeltaH_CMC(ColorLab Lab1, ColorLab Lab2)
{
    double C1 = Math.Sqrt(Math.Pow(Lab1.a, 2) + Math.Pow(Lab1.b, 2));
    double C2 = Math.Sqrt(Math.Pow(Lab2.a, 2) + Math.Pow(Lab2.b, 2));
    double H = Math.Pow(Lab1.a - Lab2.a, 2) + Math.Pow(Lab1.b - Lab2.b,
        2) - Math.Pow(C1 - C2, 2);
    if (var1 < 0) H = 0;
    else H = Math.Sqrt(H);

    return H;
}
```

# Part V.
# Data

## 23. Whitepoints

| Name | X | Y | Z | $C_x$ | $C_y$ |
|---|---|---|---|---|---|
| A | 1.09850 | 1.00000 | 0.35585 | 0.44757 | 0.40745 |
| B | 0.99072 | 1.00000 | 0.85223 | 0.34842 | 0.35161 |
| C | 0.98074 | 1.00000 | 1.18232 | 0.31006 | 0.31616 |
| D50 | 0.96422 | 1.00000 | 0.82521 | 0.34567 | 0.35850 |
| D55 | 0.95682 | 1.00000 | 0.92149 | 0.33242 | 0.34743 |
| D65 | 0.95047 | 1.00000 | 1.08883 | 0.31271 | 0.32902 |
| D75 | 0.94972 | 1.00000 | 1.22638 | 0.29902 | 0.31485 |
| E | 1.00000 | 1.00000 | 1.00000 | 0.33333 | 0.33333 |
| F2 | 0.99186 | 1.00000 | 0.67393 | 0.37208 | 0.37529 |
| F7 | 0.95041 | 1.00000 | 1.08747 | 0.31292 | 0.32933 |
| F11 | 1.00962 | 1.00000 | 0.64350 | 0.38052 | 0.37713 |

## 24. Adaption Matrix

| Name | M | | | $M^{-1}$ | | |
|---|---|---|---|---|---|---|
| XYZ Scaling | 1.0000000 | 0.0000000 | 0.0000000 | 1.0000000 | 0.0000000 | 0.0000000 |
| | 0.0000000 | 1.0000000 | 0.0000000 | 0.0000000 | 1.0000000 | 0.0000000 |
| | 0.0000000 | 0.0000000 | 1.0000000 | 0.0000000 | 0.0000000 | 1.0000000 |
| Von Kries | 0.4002400 | 0.7076000 | -0.0808100 | 1.8599364 | -1.1293816 | 0.2198974 |
| | -0.2263000 | 1.1653200 | 0.0457000 | 0.3611914 | 0.6388125 | -0.0000064 |
| | 0.0000000 | 0.0000000 | 0.9182200 | 0.0000000 | 0.0000000 | 1.0890636 |
| Bradford | 0.8951000 | 0.2664000 | -0.1614000 | 0.9869929 | -0.1470543 | 0.1599627 |
| | -0.7502000 | 1.7135000 | 0.0367000 | 0.4323053 | 0.5183603 | 0.0492912 |
| | 0.0389000 | -0.0685000 | 1.0296000 | -0.0085287 | 0.0400428 | 0.9684867 |

## 25. YCbCr Colorspaces

All spaces are from ITU-R
$C_r$ = Red Primary
$C_g$ = Green Primary
$C_b$ = Blue Primary
WP = Whitepoint

| Name | $K_R$ | $K_G$ | $K_B$ | $C_r$ | | $C_g$ | | $C_b$ | | WP |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | x | y | x | y | x | y | |
| BT.601 SD Rec. 625 | 0.299 | 0.587 | 0.114 | 0.64 | 0.33 | 0.29 | 0.6 | 0.15 | 0.06 | D65 |
| BT.601 SD Rec. 525 | 0.299 | 0.587 | 0.114 | 0.63 | 0.34 | 0.31 | 0.595 | 0.155 | 0.07 | D65 |
| BT.709 HD Rec. 1125 | 0.2126 | 0.7152 | 0.0722 | 0.64 | 0.33 | 0.3 | 0.6 | 0.15 | 0.06 | D65 |
| BT.709 HD Rec. 1250 | 0.2126 | 0.7152 | 0.0722 | 0.64 | 0.33 | 0.3 | 0.6 | 0.15 | 0.06 | D65 |

# 26. RGB Colorspaces

## 26.1. General

$C_r$ = Red Primary
$C_g$ = Green Primary
$C_b$ = Blue Primary
WP = Whitepoint

| **Name** | **Gamma** | **$C_r$** | | **$C_g$** | | **$C_b$** | | **WP** |
|---|---|---|---|---|---|---|---|---|
| | | x | y | x | y | x | y | |
| Adobe RGB | 2.2 | 0.64 | 0.33 | 0.21 | 0.71 | 0.15 | 0.06 | D65 |
| Apple RGB | 1.8 | 0.625 | 0.34 | 0.28 | 0.595 | 0.155 | 0.07 | D65 |
| Best RGB | 2.2 | 0.7347 | 0.2653 | 0.215 | 0.775 | 0.13 | 0.035 | D50 |
| Beta RGB | 2.2 | 0.6888 | 0.3112 | 0.1986 | 0.7551 | 0.1265 | 0.0352 | D50 |
| Bruce RGB | 2.2 | 0.64 | 0.33 | 0.28 | 0.65 | 0.15 | 0.06 | D65 |
| CIE RGB | 2.2 | 0.735 | 0.265 | 0.274 | 0.717 | 0.167 | 0.009 | E |
| ColorMatch RGB | 1.8 | 0.63 | 0.34 | 0.295 | 0.605 | 0.15 | 0.075 | D50 |
| Don RGB 4 | 2.2 | 0.696 | 0.3 | 0.215 | 0.765 | 0.13 | 0.035 | D50 |
| Ekta Space PS5 | 2.2 | 0.695 | 0.305 | 0.26 | 0.7 | 0.11 | 0.005 | D50 |
| NTSC RGB | 2.2 | 0.67 | 0.33 | 0.21 | 0.71 | 0.14 | 0.08 | C |
| PAL/SECAM RGB | 2.2 | 0.64 | 0.33 | 0.29 | 0.6 | 0.15 | 0.06 | D65 |
| ProPhoto RGB | 1.8 | 0.7347 | 0.2653 | 0.1596 | 0.8404 | 0.0366 | 0.0001 | D50 |
| SMPTE-C RGB | 2.2 | 0.63 | 0.34 | 0.31 | 0.595 | 0.155 | 0.07 | D65 |
| sRGB | ≈2.2 | 0.64 | 0.33 | 0.3 | 0.6 | 0.15 | 0.06 | D65 |
| Wide Gamut RGB | 2.2 | 0.735 | 0.265 | 0.115 | 0.826 | 0.157 | 0.018 | D50 |

## 26.2. Conversion Matrix

| Name | Normal | | | Inverse | | |
|---|---|---|---|---|---|---|
| sRGB | 0.4124564 | 0.3575761 | 0.1804375 | 3.2404542 | -1.5371385 | -0.4985314 |
| | 0.2126729 | 0.7151522 | 0.0721750 | -0.969266 | 1.8760108 | 0.04155600 |
| | 0.0193339 | 0.1191920 | 0.9503041 | 0.0556434 | -0.2040259 | 1.05722520 |
| NTSC RGB | 0.6068909 | 0.1735011 | 0.2003480 | 1.90999610 | -0.5324542 | -0.2882091 |
| | 0.2989164 | 0.5865990 | 0.1144845 | -0.9846663 | 1.99917100 | -0.0283082 |
| | 0.0000000 | 0.0660957 | 1.1162243 | 0.05830560 | -0.1183781 | 0.89755350 |
| Bruce RGB | 0.4674162 | 0.2944512 | 0.1886026 | 2.7454669 | -1.1358136 | -0.4350269 |
| | 0.2410115 | 0.6835475 | 0.0754410 | -0.969266 | 1.87601080 | 0.04155600 |
| | 0.0219101 | 0.0736128 | 0.9933071 | 0.0112723 | -0.1139754 | 1.01325410 |
| CIE RGB | 0.4887180 | 0.3106803 | 0.2006017 | 2.3706743 | -0.9000405 | -0.4706338 |
| | 0.1762044 | 0.8129847 | 0.0108109 | -0.513885 | 1.42530360 | 0.08858140 |
| | 0.0000000 | 0.0102048 | 0.9897952 | 0.0052982 | -0.0146949 | 1.00939680 |
| Adobe RGB | 0.5767309 | 0.1855540 | 0.1881852 | 2.0413690 | -0.5649464 | -0.3446944 |
| | 0.2973769 | 0.6273491 | 0.0752741 | -0.969266 | 1.87601080 | 0.04155600 |
| | 0.0270343 | 0.0706872 | 0.9911085 | 0.0134474 | -0.1183897 | 1.01540960 |
| Apple RGB | 0.4497288 | 0.3162486 | 0.1844926 | 2.95153730 | -1.2894116 | -0.4738445 |
| | 0.2446525 | 0.6720283 | 0.0833192 | -1.0851093 | 1.99085660 | 0.03720260 |
| | 0.0251848 | 0.1411824 | 0.9224628 | 0.08549340 | -0.2694964 | 1.09129750 |
| ProPhoto RGB | 0.7976749 | 0.1351917 | 0.0313534 | 1.3459433 | -0.2556075 | -0.0511118 |
| | 0.2880402 | 0.7118741 | 0.0000857 | -0.5445989 | 1.50816730 | 0.02053510 |
| | 0.0000000 | 0.0000000 | 0.8252100 | 0.00000000 | 0.00000000 | 1.21181280 |
| Wide Gamut RGB | 0.7161046 | 0.1009296 | 0.1471858 | 1.46280670 | -0.1840623 | -0.2743606 |
| | 0.2581874 | 0.7249378 | 0.0168748 | -0.5217933 | 1.44723810 | 0.06772270 |
| | 0.0000000 | 0.0517813 | 0.7734287 | 0.03493420 | -0.0968930 | 1.28840990 |
| Best RGB | 0.6326696 | 0.2045558 | 0.1269946 | 1.75525990 | -0.4836786 | -0.253000 |
| | 0.2284569 | 0.7373523 | 0.0341908 | -0.5441336 | 1.50687890 | 0.0215528 |
| | 0.0000000 | 0.0095142 | 0.8156958 | 0.00634670 | -0.0175761 | 1.2256959 |
| Beta RGB | 0.6712537 | 0.1745834 | 0.1183829 | 1.68322700 | -0.4282363 | -0.2360185 |
| | 0.3032726 | 0.6637861 | 0.0329413 | -0.7710229 | 1.70655710 | 0.04469000 |
| | 0.0000000 | 0.0407010 | 0.7845090 | 0.04000130 | -0.0885376 | 1.27236400 |
| ColorMatch RGB | 0.5093439 | 0.3209071 | 0.1339691 | 2.64228740 | -1.2234270 | -0.3930143 |
| | 0.2748840 | 0.6581315 | 0.0669845 | -1.1119763 | 2.05901830 | 0.01596140 |
| | 0.0242545 | 0.1087821 | 0.6921735 | 0.08216990 | -0.2807254 | 1.45598770 |
| Don RGB 4 | 0.6457711 | 0.1933511 | 0.1250978 | 1.76039020 | -0.4881198 | -0.2536126 |
| | 0.2783496 | 0.6879702 | 0.0336802 | -0.7126288 | 1.65274320 | 0.04167150 |
| | 0.0037113 | 0.0179861 | 0.8035125 | 0.00782070 | -0.0347411 | 1.24477430 |
| Ekta Space PS5 | 0.5938914 | 0.2729801 | 0.0973485 | 2.00438190 | -0.7304844 | -0.2450052 |
| | 0.2606286 | 0.7349465 | 0.0044249 | -0.7110285 | 1.62021260 | 0.07922270 |
| | 0.0000000 | 0.0419969 | 0.7832131 | 0.03812630 | -0.0868780 | 1.27254380 |
| PAL/SECAM RGB | 0.4306190 | 0.3415419 | 0.1783091 | 3.0628971 | -1.3931791 | -0.4757517 |
| | 0.2220379 | 0.7066384 | 0.0713236 | -0.969266 | 1.87601080 | 0.04155600 |
| | 0.0201853 | 0.1295504 | 0.9390944 | 0.0678775 | -0.2288548 | 1.06934900 |
| SMPTE-C RGB | 0.3935891 | 0.3652497 | 0.1916313 | 3.50539600 | -1.7394894 | -0.5439640 |
| | 0.2124132 | 0.7010437 | 0.0865432 | -1.0690722 | 1.97782450 | 0.03517220 |
| | 0.0187423 | 0.1119313 | 0.9581563 | 0.05632000 | -0.1970226 | 1.05020260 |