

# Solve system of linear equations with OpenCV: the 3 point parabola

📅 Mar 2014    👁 59089    ★ 4.8/5 (19)    💬 2

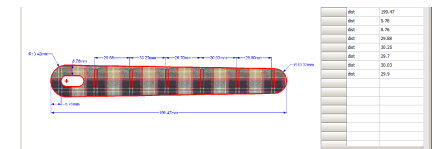
📁 OpenCV: Open Source Computer Vision Library, Software developing    🏷 Matrices, Linear algebra

*The OpenCV library is one of the most relevant open source library for Computer Vision but it contains some very useful function for linear algebra and matrix manipulation. Here is shown how to use `cv::solve` function to solve system of linear equation. As real example here is considered the case of parabola passes through 3 points.*

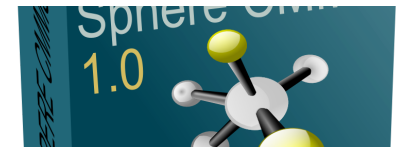
## A system of linear equations with OpenCV

Generally speaking a system of linear equations of  $n$  variables in  $m$  equation is in the form:

### In evidence



**High accuracy, wide range  
imaging measurement system  
also for soft material**



**Sphere CMMS - Software for  
Asset Maintenance**

where

- $A$  is the matrix of coefficients with  $n$  cols and  $m$  rows
- $B$  is the vector of constants with  $1$  col and  $m$  rows
- $x$  is the vector of variables with  $1$  col and  $m$  rows

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n &= b_m \end{aligned}$$

Matrix form for the above linear system is:

$$\underbrace{\begin{bmatrix} a_{11} + a_{12} + \dots + a_{1n} \\ a_{21} + a_{22} + \dots + a_{2n} \\ a_{m1} + a_{m2} + \dots + a_{mn} \end{bmatrix}}_A * \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_m \end{bmatrix}}_x = \underbrace{\begin{bmatrix} b_1 \\ b_2 \\ b_m \end{bmatrix}}_B$$

Applying this with OpenCV for the case of 3 equations with 3 variables:

```
// the matrix of coefficients
cv::Mat A = (cv::Mat_<float>(3,3) <<
             a11, a12, a13,
             a21, a22, a23,
             a31, a32, a33);
```



**Patent: A device for effective laparoscopic dilation**



**Patent for a new device for PEG**



**System for automatic disease localization and scoring in videocapsule**

This site uses technical cookies to improve user experience and analytics cookies to create site statistics. Going on visit this web site you are accepting our cookies policies. [More Info](#) [OK](#)

```
cv::Mat B = (cv::Mat_<float>(3,1) <<
            b1,
            b2,
            b3);

//the vector of variables (results)
cv::Mat x;
```

and now OpenCV solves the system:

```
cv::solve(A, B, x);
// printout the result
cout << "Result: " << x << endl;
```

A second way to solve the system is using the inverse matrix:

$$x = A^{-1} B$$

write this in OpenCV:

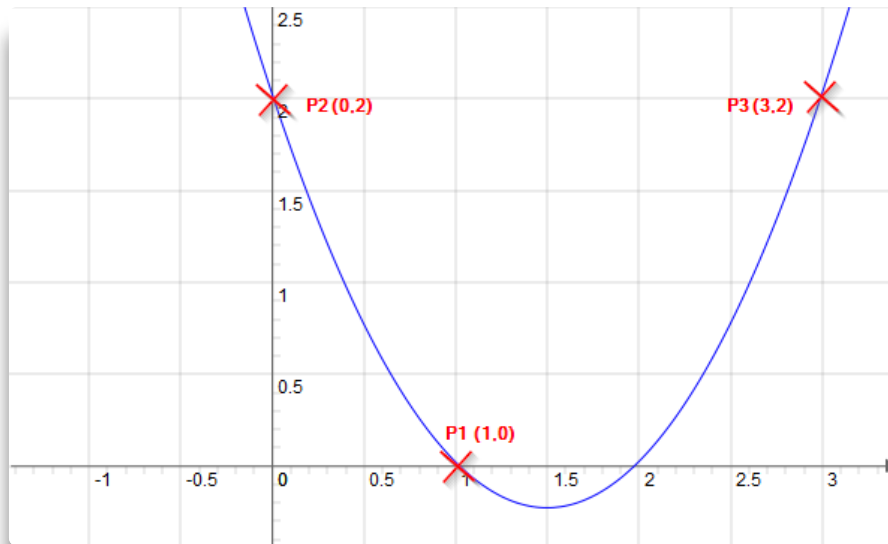
```
cv::Mat xinv = A.inv() * B;
// printout the result
cout << "Result: " << xinv << endl;
```

- [False memory leaks with OpenCV e MFC on Visual Studio](#)
- [Install OpenCV 3.2 Python/C++ on Raspberry PI](#)
- [Memory analysis on pkQueueTS as buffer of cv::Mat. Part 2, a real test.](#)
- [pkQueueTS - Treadsafe queue for OpenCV Mats](#)
- [Memory analysis on std::queue as buffer of cv::Mat. Part 1, basic tests.](#)

### Most required

- [Leadscrew system](#)
- [A Geshi plugin for CKEditor \(4.x\)](#)
- [Install OpenCV 3.2 Python/C++ on Raspberry PI](#)
- [Solve system of linear equations with OpenCV: the 3 point parabola](#)

## The case of parabola given 3 points



*A parabola passes through 3 points*

Let's go to a real case. We want to calculate the coefficient  $a, b, c$  for the equation of a parabola where 3 points are given.

General equation for the parabola is:

$$ax^2 + bx + c = y$$

The set of parabolas passes through the point  $P1(x_1, y_1)$  is:

[image or video in your own MFC interface](#)


- [Compile OpenCV 2.2 under CodeBlocks 10.5 / MinGW on Windows XP](#)
- [Standard deviation numerically stable calculation for SQL](#)
- [How to recover disappeared email from Inbox in Thunderbird](#)

Get the parabola passes through 3 points  $P_1(x_1, y_1)$ ,  $P_2(x_2, y_2)$ ,  $P_3(x_3, y_3)$  from solution to the following system:

$$\begin{cases} ax_1^2 + bx_1 + c = y_1 \\ ax_2^2 + bx_2 + c = y_2 \\ ax_3^2 + bx_3 + c = y_3 \end{cases}$$

Is this a system of linear equations ? YES !

We will use the technique as above to solve the system and find values for  $a, b, c$ .

 The coefficients for the linear system are the 3 points " $x, y$ " while the variables for the system are the parabola coefficient  $a, b, c$

Write now system for the 3 points parabola in useful form:

$$\underbrace{\begin{bmatrix} x_1^2 + x_1 + 1 \\ x_2^2 + x_2 + 1 \\ x_3^2 + x_3 + 1 \end{bmatrix}}_A * \underbrace{\begin{bmatrix} a \\ b \\ c \end{bmatrix}}_x = \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ y_m \end{bmatrix}}_B$$

1. Write the system in OpenCV for points  $P_1(1, 0)$ ,  $P_2(0, 2)$ ,  $P_3(3, 2)$  :

```
//set here your 3 points for the parabola
pt1 = cv::Point2f(1,0);
pt2 = cv::Point2f(0,2);
```

This site uses technical cookies to improve user experience and analytics cookies to create site statistics. Going on visit this web site you are accepting our cookies policies. [More Info](#) [OK](#)

```
cv::Mat A = (cv::Mat_<float>(3, 3) <<
    std::pow(pt1.x, 2), pt1.x, 1,
    std::pow(pt2.x, 2), pt2.x, 1,
    std::pow(pt3.x, 2), pt3.x, 1);

cv::Mat B = (cv::Mat_<float>(3, 1) <<
    pt1.y,
    pt2.y,
    pt3.y);

// declare a vector for results
cv::Mat abc;
```

2. Solve the system:

```
cv::solve(A, B, abc);
```

3. Printout the result:

```
cout << "Coefficients:" << endl << abc << endl;
```

returns:

```
Coefficients:
[1;
-3;
2]
```

4. Printout the equation:

This site uses technical cookies to improve user experience and analytics cookies to create site statistics. Going on visit this web site you are accepting our cookies policies. [More Info](#) [OK](#)

```
a = abc.at<float>(0);  
b = abc.at<float>(1);  
c = abc.at<float>(2);  
cout << "Equation: y = " << a << "x^2 + " << b << "x + " << c;
```

returns:

```
Equation: y = 1x^2 + -3x + 2
```

That's all. OpenCV is really nice !

## The full code for the case of 3 point parabola

```
#include <opencv2\core.hpp>  
using namespace std;  
  
int main(int argc, char *argv[])  
{  
    cv::Point2f pt1, pt2, pt3; ///< 3 points for the parabola  
    double a, b, c;           ///< Solved coefficients for the parabola equation  
  
    //set here your 3 points for the parabola  
    pt1 = cv::Point2f(1, 0);  
    pt2 = cv::Point2f(0, 2);  
    pt3 = cv::Point2f(3, 2);  
  
    // OpenCV requires the linear system in the form Ax = B  
    //  
    // the linear system for 3points parabola is  
    //
```

This site uses technical cookies to improve user experience and analytics cookies to create site statistics. Going on visit this web site you are accepting our cookies policies. [More Info](#) [OK](#)

```
//          | a(pt3.x)^2 + b(pt3.x) + c(1) = pt3.y |
// coefficients for the system are the 3 points
// variables for the system are the parabola coefficient a,b,c
//
// Finally set the matrix for the linear system solver
cv::Mat A = (cv::Mat_<float>(3, 3) <<
    std::pow(pt1.x, 2), pt1.x, 1,
    std::pow(pt2.x, 2), pt2.x, 1,
    std::pow(pt3.x, 2), pt3.x, 1);

cv::Mat B = (cv::Mat_<float>(3, 1) <<
    pt1.y,
    pt2.y,
    pt3.y);

// declare a vector for results
cv::Mat abc;

// solve the linear system
cv::solve(A, B, abc);

// printout the result
cout << "Coefficients:\n " << abc << endl;
a = abc.at<float>(0);
b = abc.at<float>(1);
c = abc.at<float>(2);
cout << "Equation:\n y = " << a << "x^2 + " << b << "x + " << c << endl;

return 0;
}
```

Vote this page: ☐ 0 ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

[Send your vote](#)



## #1 Sent by androidusr 29-04-2014

*rally easy ...opencv it's fantastic!*

## #2 Sent by asd 01-06-2017

*\ "x is the vector of variables with 1 col and m rows\ " should be n rows*

*The coding examples presented here are for illustration purposes only. The author takes no responsibility for end-user use*

*This work is property of Pk Lab. You can use it for free but you must retain author's copyright.*

[Privacy](#)[Cookie](#)[Terms and conditions](#)[Mind map](#)

This site uses technical cookies to improve user experience and analytics cookies to create site statistics. Going on visit this web site you are accepting our cookies policies. [More Info](#) [OK](#)