TECHNICAL UNIVERSITY OF CLUJ-NAPOCA

ACTA TECHNICA NAPOCENSIS

Series: Applied Mathematics, Mechanics, and Engineering
Vol. 62, Issue I, March, 2019

A JAVA CLIENT-SERVER MODEL TO SOLVE THE FORWARD AND THE INVERSE ROBOT KINEMATICS

Tiberiu Alexandru ANTAL

Abstract: *The paper describes a client-server model that can be used to solve the forward and the reverse kinematics of robot over the Internet. The client sends over the Internet a request coded in a string, the server decodes the request to find the required demand, extracts the parameters for which the solution is required, solves the request under the given conditions and sends the response back to the client. The implementation is done in the case of a 2R serial robot with two degrees of freedom along with all the numerical algorithms required to obtain the proposed solutions. Results can be viewed numerically or graphically by interfacing Java with AutoCAD and Canvas X.*

Key words: *client, forward kinematics, inverse kinematics, java, robot, server.*

1. INTRODUCTION

Java is an object-oriented programming language developed by the Green Team made up of James Gosling, Mike Sheridan, and Patrick Naughton, from Sun Microsystems (which is now a subsidiary of Oracle Corporation), as a part of a project called Green in June 1991. Initially, called “Greentalk” by James Gosling, the language was renamed to “Oak” after the tree that was planted outside of James Gosling’s office window. In 1995, “Oak” was renamed as "Java" because it was already a trademark by Oak Technologies [1]. The first demonstration of the language involved building a PDA (Personal Digital Assistant) prototype called Star7 that was able to manipulate the VCR in order to obtain some interactivity with the TV set connected to de VCR. Star 7 had a GUI (Graphical User Interface) and a smart agent called "Duke" to assist the user. Since then Java went through may changes, leaving aside the technical evolution, some of these were rebranding and renaming. Today, OpenJDK - released under license GPL v2 - is an open source implementation of the Java Standard Edition platform with contributions from Oracle and the open Java community while Oracle JDK - previously called Sun JDK - is licensed under Oracle Binary Code License Agreement. The cost of a commercial Oracle JDK license can be obtained only by contacting oracle, while OpenJDK is completely free and can be used in accordance with the GPL v2 license. Despite of the licensing differences the main features were kept the same so the simple developer could choose easily to recompile his code under the desired license. The robustness and flexibility of language has, in time, attracted programmers from all areas, including mechanics, as Java could be used from building portable graphical interfaces [2], [4], [5], interact with CAD software like AutoCAD [3] under certain restrictions, communicate with microcontrollers who run different processes or drive robots [6] - [10], [18] - [21] connect to databases [11], [12], or to solve numerically nonlinear equation or systems of equations specific the mechanical field [13], [14]. One area with huge application in robotics is the client-server applications [8], [9] where Java excels due to specialized packages over the Internet communication.

2. THE CLIENT-SERVER APPROACH IN THE FORWARD AND INVERSE KINEMATICS PROBLEM

The client server application involves remote communication between the client and the server using the public Internet network. The server will provide solutions to the forward and to the invers kinematics of the robot, while the client will issue the calls for the resolution of the forward or invers kinematic problem. At this stage the type of the robot is not important what counts is the information that needs to be placed in a request for the server to handle the request. Forward kinematics is based on the kinematic equations of a robot and computes the position of the end-effector for given values of the joint parameters. The reverse process that computes the joint parameters for a given position of the end-effector is known as inverse kinematics. This means that the client must use some code for the forward kinematics followed by the set of the parameters necessary to solve the kinematic equations. To request the reverse kinematics solution a different code must be used followed by the end-effector position. Considering a 2R (two rotations) dof (degree of freedom) planar robot the two 'kinematic' problems are shown in Figure 1 and Figure 2.

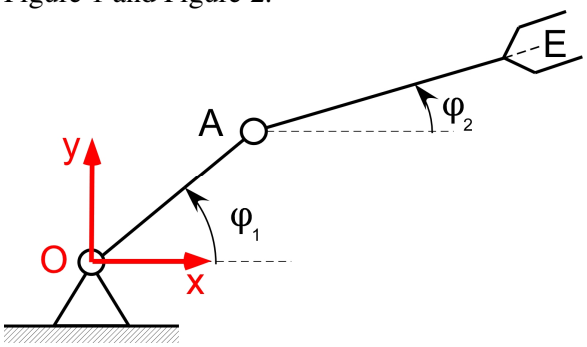

Fig. 1. – Forward kinematic model for a 2 dof planar robot.

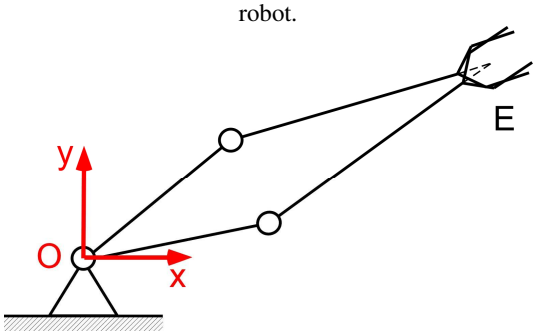

Fig. 2. – Inverse kinematic model for a 2 dof planar robot.

The forward kinematic problem is simple being based on the following equations:

$$\begin{cases} x_E = l_{OA}\cos(\varphi_1) + l_{AE}\cos(\varphi_2) \\ y_E = l_{OA}\sin(\varphi_1) + l_{AE}\sin(\varphi_2) \end{cases} \quad (1)$$

If l_{OA} and l_{AE} lengths of the links are known, for a given set of $\{\varphi_1, \varphi_2\}$ the coordinates (x_E, y_E) of the end-effector are computed directly from (1). The invers kinematical model is not that simple, as Figure 2 shows, two solutions are possible. No special mathematics is needed to solve this selection between the two solutions however, in the following, an iterative numerical method will be used to solve this system of non-linear equations. The scientific literature describes the NR (Newton-Raphson) method as being one of the most widely used for finding solutions of non-linear systems of equations. Compared to the general NR case, in this work, the method is customized for the concrete situation of the robot in Figure 1. Consider the problem of determining the solutions (α, β) of the following system of equations:

$$\begin{cases} f(\varphi_1, \varphi_2) = 0 \\ g(\varphi_1, \varphi_2) = 0 \end{cases} \quad (2)$$

The NR method will create two series $(\varphi_1)_n$ and $(\varphi_2)_n$ starting from an initial point $\{(\varphi_1)_0, (\varphi_2)_0\}$ that will which will converge to (α, β). The relationships for the definitions of these series are recurrent and have the following form:

$$(\varphi_1)_{n+1} = (\varphi_1)_n - \frac{f \cdot g'_{\varphi_2} - g \cdot f'_{\varphi_2}}{\Delta} \quad (3)$$

and

$$(\varphi_2)_{n+1} = (\varphi_2)_n - \frac{g \cdot f'_{\varphi_1} - f \cdot g'_{\varphi_1}}{\Delta} \quad (4)$$

where:

$$\Delta = f'_{\varphi_1} \cdot g'_{\varphi_2} - f'_{\varphi_2} \cdot g'_{\varphi_1} \quad (5)$$

The following notations were used for partial derivatives:

$$f'_{\varphi_1} = \frac{\partial f}{\partial \varphi_1} \quad (5)$$

$$f'_{\varphi_2} = \frac{\partial f}{\partial \varphi_2} \quad (6)$$

$$g'_{\varphi_1} = \frac{\partial g}{\partial \varphi_1} \quad (7)$$

$$g'_{\varphi_2} = \frac{\partial g}{\partial \varphi_2} \quad (8)$$

The recursive procedure used to generate the (3) and (4) terms of the numerical solution series should be stopped when the amount from (9)

$$|(\varphi_1)_{n+1} - (\varphi_1)_n| + |(\varphi_2)_{n+1} - (\varphi_2)_n| \quad (9)$$

will be less than a fixed maximum error (usually marked with ε). At that moment it can be said that $(\varphi_1)_{n+1}$ and $(\varphi_2)_{n+1}$ are the approximate solution of the system (2). The numerical approximate derivatives of the formulas (5) to (8) will be calculated as follows:

$$f'_{\varphi_1}(\varphi_1, \varphi_2) = \frac{f(\varphi_1 + h, \varphi_2) - f(\varphi_1 - h, \varphi_2)}{2h} \quad (10)$$

$$f'_{\varphi_2}(\varphi_1, \varphi_2) = \frac{f(\varphi_1, \varphi_2 + h) - f(\varphi_1, \varphi_2 - h)}{2h} \quad (11)$$

$$g'_{\varphi_1}(\varphi_1, \varphi_2) = \frac{g(\varphi_1 + h, \varphi_2) - g(\varphi_1 - h, \varphi_2)}{2h} \quad (12)$$

$$g'_{\varphi_2}(\varphi_1, \varphi_2) = \frac{g(\varphi_1, \varphi_2 + h) - g(\varphi_1, \varphi_2 - h)}{2h} \quad (13)$$

3. JAVA CODING AND PARAMETERS USED FOR THE FORWARD AND INVERSE KINEMATICS FOR THE 2 DOF PLANAR ROBOT

The client will send strings to the server using the following syntax:

- for the direct kinematics: DK φ_1, φ_2.
- for the inverse kinematics: RK $x_E, y_E, (\varphi_1)_0, (\varphi_2)_0$

The string will be parsed at the server by the following method:

```
void parseString(String s) {
  String k;
  double a, b, f10, f20;
  StringTokenizer tk = new
StringTokenizer(s, " , ");
  try {
    k = tk.nextToken();
    a =
Double.parseDouble(tk.nextToken());
    b =
Double.parseDouble(tk.nextToken());

    if (k.equals("DK")) {
      f1 = Math.toRadians(a);
      f2 = Math.toRadians(b);
      dirK();
    }
    if (k.equals("RK")) {
      xe = a;
      ye = b;
      f10 =
Double.parseDouble(tk.nextToken());
      f20 =
Double.parseDouble(tk.nextToken());
      invK(f10, f20);
    }
  }
  catch (Exception e) {
    System.out.println("Parse error ->
" + e);
  }
}
```

With the help of the `StringTokenizer` class the kinematics coding and the parameters are extracted and the corresponding method is called to solve the problem. For the direct kinematics the method is:

```
void dirK() {
  xe = l1 * Math.cos(f1) + l2 *
Math.cos(f2);
  ye = l1 * Math.sin(f1) + l2 *
Math.sin(f2);
}
```

For the invers kinematics the method is:

```
public void invK(double xk0, double
yk0) {
  double xk1, yk1, xk, yk, aux1, aux2;
  xk = Math.toRadians(xk0);
  yk = Math.toRadians(yk0);
  do {
    aux1=(F(xk, yk) * Gd1f2(xk, yk) -
G(xk, yk) * Fd1f2(xk, yk)) / delta(xk,
yk);
    aux2 = (G(xk, yk) * Fd1f1(xk, yk) -
F(xk, yk) * Gd1f1(xk, yk)) / delta(xk,
yk);
    xk1 = xk - aux1;
    yk1 = yk - aux2;
```

```
    xk = xk1;
    yk = yk1;
  }     while     (Math.abs(aux1)     +
Math.abs(aux2) > eps);
  f1 = xk1;
  f2 = yk1;
}
```

For the sake of simplicity, the above-provided code omits the safety measures that should stop it after a predefined maximum number of iterations if the required convergence conditions are not met.

The calculation of partial derivatives is based on the following methods:

```
public double F(double f1, double f2) {
  return  l1  *  Math.cos(f1)  +  l2  *
Math.cos(f2) - xe;
}
public double Fd1f1(double f1, double
f2) {
  return (F(f1 + h, f2) - F(f1 - h, f2))
/ 2. / h;
}
public double Fd1f2(double f1, double
f2) {
  return (F(f1, f2 + h) - F(f1, f2 -
h)) / 2. / h;
}
```

The above methods are given for the function *f* from (2) for the function *g* the methods can be deduced easily based on (11) and (12).

4. PRESENTATION OF THE RESULTS

The results obtained can be presented directly in Java by creating classes to simulate the robot operation. However, in this work, I preferred to make the connection between Java, Canvas X and AutoCAD. Based on the scripting technologies given in (15) and (16), (17) the computed positions of the 2 dof planar robot can be stored as files. AutoCAD is a well-known CAD software while Canvas X is an illustration software with good scripting capabilities. If the system of axes in AutoCAD is the one used in mathematics and engineering, the one in canvas is specific to illustration programs, so the Oy axis grows down. Consider the following piece of Java code to output the results, where `autocad` and `canvas` are boolean variables used to select one of the two possible output cases:

```
if (autocad) {
  System.out.println("pline");
  System.out.printf("0.0,0.0\n%5.4f,
%5.4f\n", xa,ya);
  System.out.printf("%5.4f,%5.4f\n\n",
xe, ye);
}
if (canvas) {
  System.out.print("aly.CreateLine ");
   System.out.printf("%5d,%5d,%5.4f,
%5.4f\n", tx,ty, xa+tx, ya+ty);
  System.out.print("aly.CreateLine ");

System.out.printf("%5.4f,%5.4f,%5.4f,
%5.4f\n", xa+tx, ya+ty, xe+tx, ye+ty);
}
```

AutoCAD can be programmed very simply using script files [16]. A script file is a text file that has the *scr* extension and contains AutoCAD commands. If the `autocad` variable is set to *true* the output will be formatted for AutoCAD. The *pline* command is used to draw the robot from the example using two lines (OA and AE segments), for each computed position. If the `canvas` variable is set to *true* the output will be formatted for CanvasX. Canvas X ca be programmed in VBScript and the Java code will create two lines (one line is created with the `CreateLine` statement) for each computed position of the robot. These must be stored in a text file with the *vbs* extension and loaded into Canvas X as described in (15) and included at the right position in the *vbs* file as shown in the following piece of code:

```
' Declare variables and constants.
Set                cv                =
GetObject(,"Canvas.Application")
Set      doc      =      cv.Documents.
Add(cvsIllustrationDocument)
Set aly = cv.ActiveDocument.ActivePage.
ActiveLayer

' aly.CreateLine statements must be
' inserted here
aly.CreateLine 15,15,6.72,9.38
aly.CreateLine 6.72,9.38,16.00,17.00
' first position above
aly.CreateLine 15,15,5.10,16.43
aly.CreateLine 5.10,16.43,17.00,18.00
' second position above …
doc.Selection.SelectAll
Set grp = doc.Selection.MakeGroup
grp.DrawObject.Flip cvsVertical
```

If the client sends to the server requests, using the `serverRQ()` method, a set *k* of points on a line (`y=x+1`) as shown in the following code, the server response will be that from Figure 3.

```
for (x = 1.; x <= 15.; ++x) {
  y = x + 1.; //line equation
  serverRQ("RK " + x + "," + y +
",90.,10.");
}
```

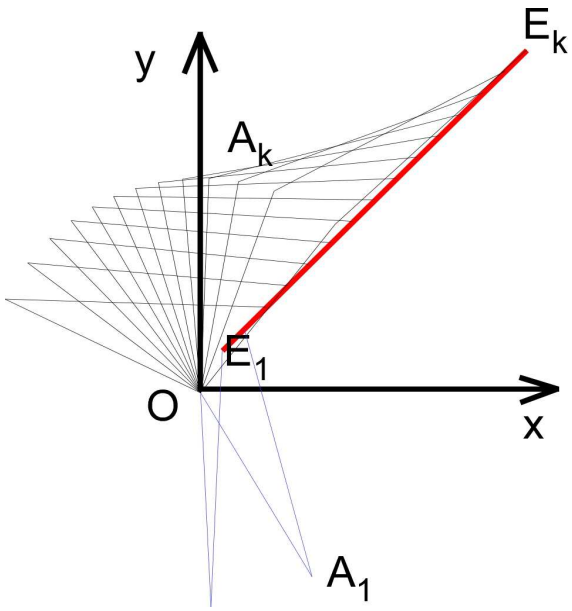

Fig. 3. – Response of the server at the inverse kinematic request for a set of points on a line.

As shown in Figure 3, although all E points are on the line, the first two A points (A_1 and A_2) are not consecutive to the rest of the points A_k of the solution. This situation is normal because the NR numerical method used is iterative and the obtained solutions are depending on the chosen starting points. As these staring points are transmitted as parameters to the NR solver in the request, the client should change the staring points from 90,10 to 150,0 as shown in the following code in order to obtain consecutive A points:

```
for (x = 1.; x <= 15.; ++x) {
  y = x + 1.;
  if (x > 3.)
    serverRQ("RK " + x + "," + y +
",90.,10.");
  else
    serverRQ("RK " + x + "," + y +
",150.,0.");
}
```

5. REFERENCES

[1] https://www.javatpoint.com/history-of-java.

[2] ANTAL, T. A., *GUI's in JDeveloper*, Acta Technica Napocensis, Series: Applied Mathemathics and Mechanics, Nr. 52, Vol. IV, 2009, p.27-32, ISSN 1221-5872.

[3] ANTAL, T .A., *Programming AutoCAD using JAWIN from Java in JDeveloper*, Acta Technica Napocensis, Series: Applied Mathemathics and Mechanics, Nr. 53, Vol. III, 2010, p.481-486, ISSN 1221-5872.

[4] ANTAL, T. A., *Elemente de Java cu JDeveloper - îndrumător de laborator*, Editura UTPRES, 2013, p.150, ISBN: 978-973-662-827-6.

[5] ANTAL, T. A., *Java - Iniţiere - îndrumător de laborator*, Editura UTPRES, 2013, p. 246, ISBN: 978-973-662-832-0.

[6] ANTAL, Tiberiu Alexandru. *Raspebrry Pi 3 programming, in java, using Blue J and JDeveloper based on Pi4J*. Acta Technica Napocensis - Series: Applied Mathematics, Mechanics and Engineering, [S.l.], v. 60, n. 1, p. 13-18. 2017. ISSN 1221-5872.

[7] ANTAL, Tiberiu Alexandru. *Arduino Leonardo programming under Windows, in Java, from JDeveloper using Ardulink*. Acta Technica Napocensis - Series: Applied Mathematics, Mechanics and Engineering, [S.l.], v. 60, n. 1, p. 7-12. 2017. ISSN 1221-5872.

[8] ANTAL, Tiberiu Alexandru; CHELARU, Julieta Daniela. *A multithreaded java client-server model for robot interaction.* Acta Technica Napocensis - Series: Applied Mathematics, Mechanics and Engineering, [S.l.], v. 60, n. 3, p. 331-336. sep. 2017. ISSN 1221-5872.

[9] ANTAL, Tiberiu Alexandru. *Considerations on the serial PC - Arduino Uno R3 interaction, in Java, using JDeveloper, for a 3R serial robot, based on the Ardulink library.* Acta Technica Napocensis - Series: Applied Mathematics, Mechanics and Engineering,, [S.l.], v. 61, n. 1, p. 7-10. mar. 2018. ISSN 1221-5872.

[10] ANTAL, Tiberiu Alexandru. *3R serial robot control based on Arduino/Genuino Uno, in Java, using JDeveloper and Ardulink.*

Acta Technica Napocensis - Series: Applied Mathematics, Mechanics and Engineering,, [S.l.], v. 61, n. 1, p. 11-15. mar. 2018. ISSN 1221-5872.

[11] ANTAL, T. A., *ACCESS to an ORACLE DATABASE using JDBC*, Acta Technica Napocensis, Series: Applied Mathemathics and Mechanics, Nr. 47, Vol. III, 2004, p.63-68, ISSN 1221-5872.

[12] ANTAL, T. A., *Utilization of the relational databases in the storage and retrieval of mechanisms.* Acta Technica Napocensis, Series: Applied Mathemathics and Mechanics, Nr. 47, Vol. IV, 2004, p.7-14, ISSN 1221-5872.

[13] ANTAL, T .A., *An object oriented implementation for the study of the sliding equalization, at the points where the gearing starts and ends*, Acta Technica Napocensis, Series: Applied Mathemathics and Mechanics, Nr. 50, Vol. I, 2007, p.33-38, ISSN 1221-5872.

[14] ANTAL, T .A., ANTAL, A., *An object oriented model for the study of the specific addendum modifications based on the sliding and relative velocities equalization at helical gears*, Acta Technica Napocensis, Series: Applied Mathemathics and Mechanics, Nr. 51, Vol. II, 2008, p.45-50, ISSN 1221-5872.

[15] ANTAL, T.A., *Programming Canvas X Pro 16 using scripting technologies, acta technica napocensis*, Acta Technica Napocensis, Series: Applied Mathematics, Mechanics, and Engineering, Vol. 58, Issue II, 2015, p.151-156, ISSN 1221-5872.

[16] TIUCA, T., PRECUP, T., ANTAL, T., *Dezvoltarea aplicaţiilor cu AutoCAD şi AutoLISP*, Editura Promedia Plus Computers, 1995, p. 303, ISBN 973-96862-2-2.

[17] ANTAL, T. A., *Mechanism displacement analysis with AutoLisp in AutoCAD*, Series: Applied Mathematics and Mechanics, Nr. 45, 2002, U. T. PRESS - The Technical University form Cluj-Napoca, Romania, p.19-24, ISSN 1221-5872.

[18] Detesan, OA, *The geometric and kinematic model of rttrr small-sized modular robot*, Acta Technica Napocensis Series-Applied Mathematics Mechanics And Engineering. Volume: 58, Issue: 4, Pages: 513-518, 2015, ISSN: 1221-5872

[19] Detesan, OA, *The numerical simulation of trr small-sized robot*, Acta Technica Napocensis Series-Applied Mathematics Mechanics And Engineering, Volume: 58, Issue: 4, Pages: 519-524, 2015, ISSN: 1221-5872.

[20] Aurora Felicia Cristea, Carmen Monica BĂLCĂU, Dorian Cosmin Deac - *Studies and researches regarding the calculation of resistance and designing the vibrating plate/table „MEVI”* - ACTA Technica Napocensis, Series: Applied Mathematics and Mechanics, 2017, vol II, nr. 60, pag. 235-238, ISSN 1221-5872.

[21] Aurora Felicia Cristea - *Calculation of foundation screw regarding the vibrating table „MEVI”* - ACTA Technica Napocensis, Series: Applied Mathematics and Mechanics, 2017, vol III, nr. 60, pag.389-392, ISSN 1221-5872.

UN MODEL JAVA, CLIENT-SERVER, PENTRU SOLUTIONAREA PROBLEMEI CINEMATICII DIRECTE SI INVERSE A UNUI ROBOT

Lucrarea descrie un model de aplicaţie Java, client-server, care poate fi folosit pentru a rezolva problema cinematică directă şi inversă a unui robot prin Internet. Clientul transmite o cerere codificată sub forma unui şir de caractere, serverul decodifică cererea pentru a determina operaţia de efectuat, apoi extrage parametrii necesari pentru a gasi soluția, rezolvă cererea în condițiile date și trimite răspunsul înapoi clientului tot sub forma unui şir. Implementarea este realizată în cazul unui robot serial 2R cu două grade de libertate împreună cu toţi algoritmii numerici necesari soluţionării problemelor propuse. Rezultatele se pot vizualiza numeric sau grafic prin interfaţarea coului Java cu produsele software AutoCAD și Canvas X.

ANTAL Tiberiu Alexandru, Professor, dr. eng., Technical University of Cluj-Napoca, Department of Mechanical System Engineering, antaljr@bavaria.utcluj.ro, 0264-401667, B-dul Muncii, Nr. 103-105, Cluj-Napoca, ROMANIA.