



TECHNICAL UNIVERSITY OF CLUJ-NAPOCA

ACTA TECHNICA NAPOCENSIS

Series: Applied Mathematics, Mechanics, and Engineering
Vol. 67, Issue I, March, 2024

GENERALIZATION BY PARAMETERIZATION WITH ASSOCIATED ARRAYS, IN PHP, IN A MANIPULATOR COMPUTATION

Tiberiu Alexandru ANTAL

***Abstract:** The paper approached, using the PHP open-source server-side scripting language, a case of abstraction for a 2R manipulator using routines and parameters. If certain design and abstraction conditions are met general routines can be built to allow code reuse. However, these computational libraries will only cover a limited domain for manipulators as finding the right balance between abstraction and simplicity is influenced by generality, context and dependencies.*

***Key words:** function, manipulator, parameter, PHP, reuse, routine.*

1. INTRODUCTION

Hypertext Preprocessor abbreviated as PHP is an open-source, server-side scripting language designed initially for web purpose. Born in Greenland, grew up in Denmark and Canada, Rasmus Lerdorf, developed PHP, in 1994, while working on his personal website. Originally PHP meant "Personal Home Page" as it was a tool to manage and maintain Rasmus Lerdorf's personal website. Over time PHP has evolved into a solid server-side scripting language for web development. Today, the name PHP is redefined as the pun "PHP = Hypertext Preprocessor" to emphasize the power of the language in processing hypertext and creating dynamic web pages. As described in [1] the idea of the language is to program without effort. Being a weak typed language [2], with a simplified syntax, where variable types are determined dynamically during runtime beginners start learning and working in the language easier [11], [12], [13]. There was another language in the history of programming that started from the idea of learning as quickly as possible to code. The original creators of BASIC wanted to create a version of the Fortran language that was easier to learn. The resulted language had the acronym BASIC from Beginner's All-purpose Symbolic Instruction

Code and was developed at Dartmouth College in 1964 by Prof. John G. Kemeny and Prof. Thomas E. Kurtz. Over the years, numerous dialects or variations of BASIC emerged. Microsoft introduced Visual Basic in 1991, providing a graphical development environment for building Windows applications [4] that could be used to perform complex numerical calculations [9], [10]. In time, BASIC became one of the scripting languages for system administration tasks of the Windows Operating System under the name of VBScript as well as server-side scripting language of the IIS (Internet Information Services) the web server [8] created by Microsoft for use with the Windows Server operating systems. Without going into details, the PHP development environment can be used independently or integrated as a module in the Apache web server, is open-source and cross-platform and can run on various operating systems. This is probably the reason for the immense success enjoyed by the language. An excellent open-source text editor that can be used to edit and run PHP is the Atom [5] text editor developed by GitHub. This can be customized to edit, debug and run, using plugins, the PHP code outside the PHP development environment (see Figure 1).

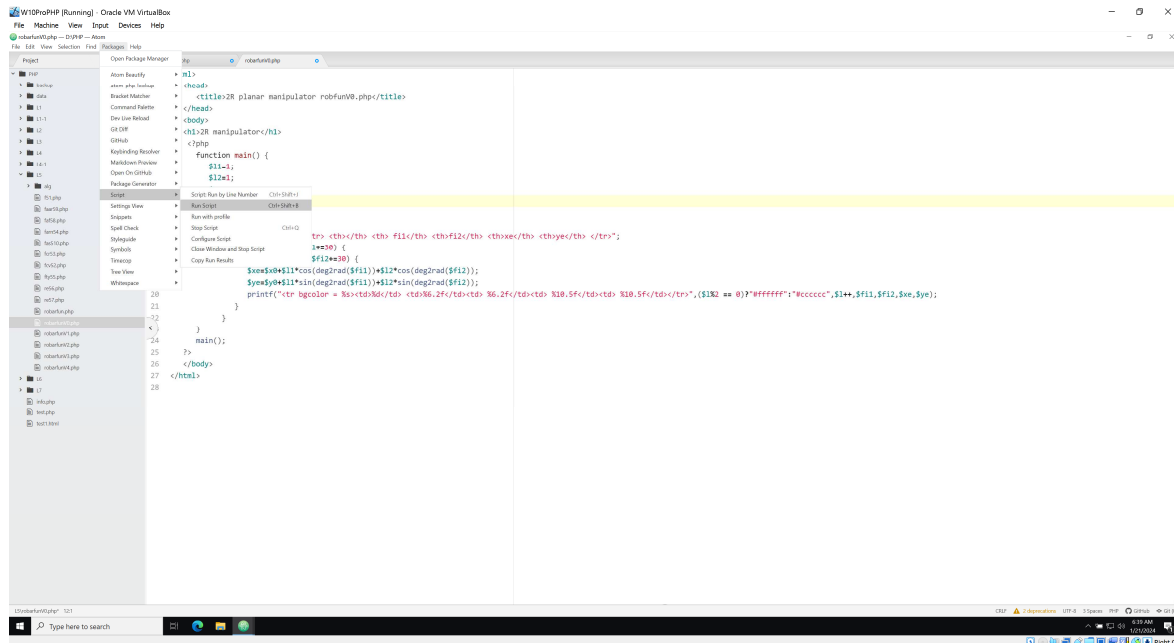


Fig. 1. - The Atom editor configured to run PHP code.

1.1 Some words on the computations to be carried out

Consider the two link planar manipulator with revolute joints from Figure 2.

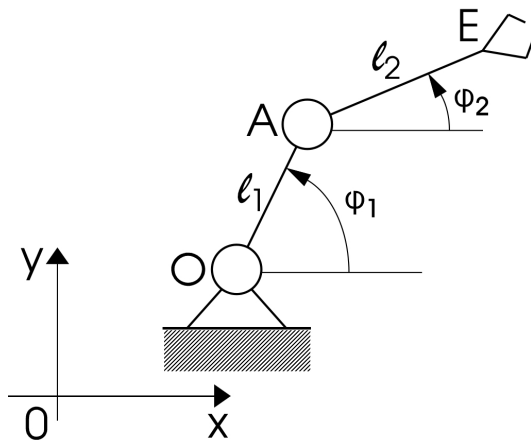


Fig. 2. - The two-link with two revolute joints planar manipulator.

We are interested to compute the coordinates of the E point having as inputs the φ_1 and φ_2 angles. The Cartesian position of the end-effector, in E, of the manipulator in terms of the input angles with respect of x_0y_0 is:

$$\begin{cases} x_E = x_0 + l_1 \cos(\varphi_1) + l_2 \cos(\varphi_2) \\ y_E = y_0 + l_1 \sin(\varphi_1) + l_2 \sin(\varphi_2) \end{cases} \quad (1)$$

1.2 Structured code for the PHP environment

The following code is written to produce the results as a HTML table. The code runs on the Apache web server (as part of the PHP environment) and the results are shown in a browser (Microsoft Edge). The structured programming paradigm is used to compute the results from Figure 3.

```
<html>
<head>
  <title>2R      planar      manipulator
robfunV0.php</title>
</head>
<body>
  <h1>2R manipulator</h1>

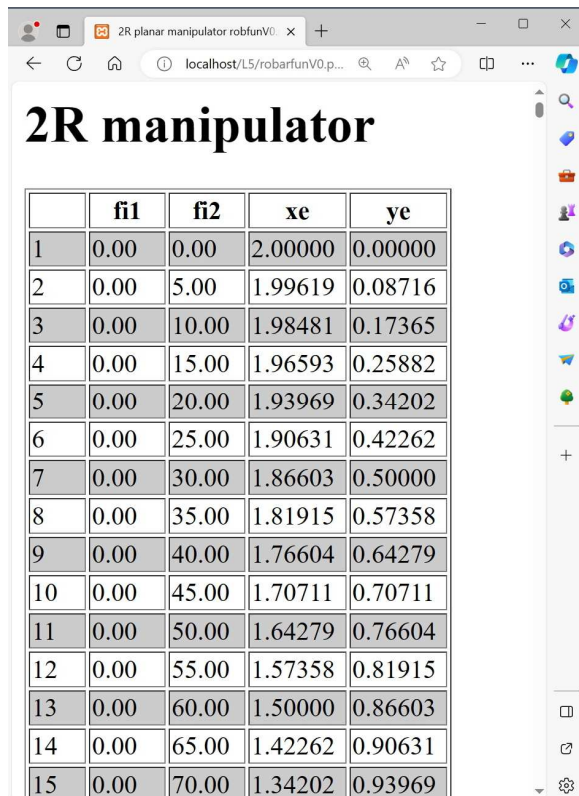
  <?php
  function main() {
    $l1=1;
    $l2=1;
    $x0=0;
    $y0=0;

    $l1=1;
    echo  "<table border=1>  <tr>
  <th></th> <th> fi1</th> <th>fi2</th>
  <th>xe</th> <th>ye</th> </tr>";
    for($fi1=0;$fi1<=360;$fi1+=5) {
      for($fi2=0;$fi2<=360;$fi2+=5) {
        $xe=$x0+$l1*cos(deg2rad($fi1))+
        $l2*cos(deg2rad($fi2));
        $ye=$y0+$l1*sin(deg2rad($fi1))+
        $l2*sin(deg2rad($fi2));
```

```

        printf("<tr        bgcolor        =
%s><td>%d</td>        <td>%6.2f</td><td>
%6.2f</td><td>        %10.5f</td><td>
%10.5f</td></tr>", ($l%2        ==
0)?"#ffffff":"#cccccc", $l++, $fi1, $fi2,
$xe, $ye);
    }
}
main();
?>
</body>
</html>

```



	fi1	fi2	xe	ye
1	0.00	0.00	2.00000	0.00000
2	0.00	5.00	1.99619	0.08716
3	0.00	10.00	1.98481	0.17365
4	0.00	15.00	1.96593	0.25882
5	0.00	20.00	1.93969	0.34202
6	0.00	25.00	1.90631	0.42262
7	0.00	30.00	1.86603	0.50000
8	0.00	35.00	1.81915	0.57358
9	0.00	40.00	1.76604	0.64279
10	0.00	45.00	1.70711	0.70711
11	0.00	50.00	1.64279	0.76604
12	0.00	55.00	1.57358	0.81915
13	0.00	60.00	1.50000	0.86603
14	0.00	65.00	1.42262	0.90631
15	0.00	70.00	1.34202	0.93969

Fig. 3. - HTML results of the structured PHP code used to compute the coordinates of the E point with respect of the input angles.

1.3 Structured code for Atom

Eliminating HTML tags from the previous code a shorter and more readable code is obtained that will run directly in Atom if the marked script from Figure 4 is installed and enabled.

And the result from Figure 5 is printed directly in the output window of the Atom editor. The php code markup must be preserved, that is the code must be written between `<?php` and `?>` tags.

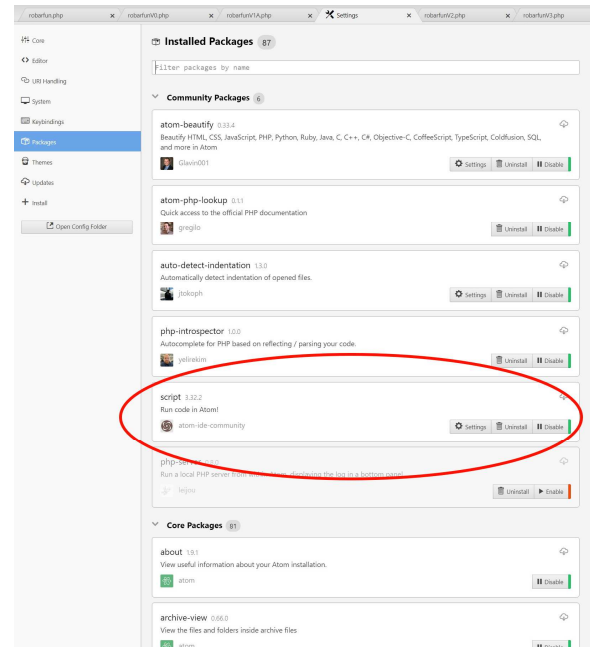


Fig. 4. - Atom script to run PHP code directly in the editor.

The corresponding code follows:

```

<?php
function main() {
    $l1=1;
    $l2=1;
    $x0=0;
    $y0=0;
    printf("        fi1        fi2        xe
ye\n");
    $l=1;
    for($fi1=0;$fi1<=360;$fi1+=30) {
        for($fi2=0;$fi2<=360;$fi2+=30) {
            $xe=$x0+$l1*cos(deg2rad($fi1))+
$l2*cos(deg2rad($fi2));
            $ye=$y0+$l1*sin(deg2rad($fi1))+
$l2*sin(deg2rad($fi2));
            printf("%3d) %6.2f %6.2f %9.5f
%9.5f\n", $l++, $fi1, $fi2, $xe, $ye);
        }
    }
    main();
?>

```

If the Apache server is up the code can also be executed inside a browser, however the displayed results will look like a bunch of numbers without any alignment or formatting, because the browser's corresponding arrangement tags in the html has been cut out from the code (the code produces the data without any formatting).

	fi1	fi2	xe	ye
1	0.00	0.00	2.00000	0.00000
2	0.00	30.00	1.86603	0.50000
3	0.00	60.00	1.50000	0.86603
4	0.00	90.00	1.00000	1.00000
5	0.00	120.00	0.50000	0.86603
6	0.00	150.00	0.13397	0.50000
7	0.00	180.00	0.00000	0.00000
8	0.00	210.00	0.13397	-0.50000
9	0.00	240.00	0.50000	-0.86603
10	0.00	270.00	1.00000	-1.00000
11	0.00	300.00	1.50000	-0.86603
12	0.00	330.00	1.86603	-0.50000
13	0.00	360.00	2.00000	-0.00000
14	30.00	0.00	1.86603	0.50000
15	30.00	30.00	1.73205	1.00000
16	30.00	60.00	1.36603	1.36603
17	30.00	90.00	0.86603	1.50000
18	30.00	120.00	0.36603	1.36603
19	30.00	150.00	0.00000	1.00000
20	30.00	180.00	-0.13397	0.50000
21	30.00	210.00	0.00000	-0.00000
22	30.00	240.00	0.36603	-0.36603
23	30.00	270.00	0.86603	-0.50000
24	30.00	300.00	1.36603	-0.36603
25	30.00	330.00	1.73205	-0.00000
26	30.00	360.00	1.86603	0.50000
27	60.00	0.00	1.50000	0.86603

Fig. 5. - Direct PHP results, in Atom, of the structured code used to compute the coordinates of the E point with respect of the input angles.

2. REUSABILITY

Modular programming was one of the dedicated ways of constructing programs from small units of code [6]. The common method to reach such a code organization is based on decomposability, which lead to the design method named top-down design. The designer starts with the most abstract description of the system and then this is refined, by decomposition, to simpler subsystems until they are close enough to allow direct implementation by routines. Reusability is obtained then by building libraries of routines. The routine is a software unit that may be called to execute a task based on certain inputs and producing certain outputs and possibly modifying some data other data. The terms of subroutine, subprogram, procedure or function are sometimes used instead of routine. The name procedure is used

for a routine that does not return a result while that of function for a routine that returns one result.

2.1 The concept of routine in PHP

In PHP the routine is implemented under the term of function (as in C [7]), and it is a unit of code with a given name (the name of the function) possibly working with some input data (the parameters) and possibly returning a single value (the output). One way of reorganizing the presented structured code is to decompose the grouped computation and printing tasks into distinct subsystems, one for computation and one for printing. This means that all computations must be stored using arrays. Scalar types can store a single value; array types can store more values in a single variable. PHP supports two kinds of arrays: indexed and associative. Indexed array use keys that are integers (starting from 0) to identify the position in the array. Associative arrays use keys that are strings to identify elements in the array. The following code is using three global indexed arrays to store the data ($\$x_{ea}$ - array to store x_E from (1); $\$y_{ea}$ - array to store y_E from (1); $\$e_a$ - array that combines the two array to another array to hold under a single name both distinct arrays). The `global` keyword (see `compute()` and `printA()`) must be used in the function in order to access the global variable.

```
<?php
    $xea = array();
    $yea = array();
    $ea = array();

    function xe($x0,$l1,$l2,$fi1,$fi2) {
        return
        $x0+$l1*cos(deg2rad($fi1))+$l2*cos(deg
        2rad($fi2));
    }

    function ye($y0,$l1,$l2,$fi1,$fi2) {
        return
        $y0+$l1*sin(deg2rad($fi1))+$l2*sin(deg
        2rad($fi2));
    }

    function compute() {
        global $xea, $yea, $ea;
        $l1=1;$l2=1;
        $x0=0;$y0=0;

        for($fi1=0;$fi1<=360;$fi1+=120) {
```

```

    for($fi2=0;$fi2<=360;$fi2+=120) {
        $xea[] = xe($x0,$l1,$l2,$fi1,$fi2);
        $yea[] = ye($y0,$l1,$l2,$fi1,$fi2);
        $sea = [$xea, $yea];
    }
}

function printA() {
    global $sea;
    [$xea, $yea] = $sea;
    $l=1; $i=0;
    printf("        xe            ye\n");
    foreach($xea as $x)
        printf("%3d                %10.5f
%10.5f\n", $l++, $x, $yea[$i++]);
}

function main() {
    compute();
    printA();
}
main();
?>

```

The `main()` function only contains two calls, one to `compute()`, and one to `printA()`. It is very clear that the readability of the code has increased and that the separation into two distinct code units is possible, although instead of parameters, global array variables were used

2.2 Routine generalization in PHP

Once the decomposition has been made into simpler and directly implementable tasks, the next stage is to give up global variables, which can be modified by anyone and at any time, and move to parameters that appear as arguments in the call and, depending on the context and the nature of the call, by value or by reference, migrates input and output to the code that processes them. This approach works to build a library of more general routines if:

- a subtask can be identified and extracted as a routine with a small number of inputs and outputs - simple task with few inputs/outputs;
- the subtask can be completely isolated from others, thus eliminating any common parts between distinct subtask - no commonality or other interferences with other subtasks;
- inputs/outputs with no complex structures are involved in data transfers

as this will compromise the independence of the isolated task due to the specific adaptation needs in the processing of the complex structured input/output parameter;

The previous code is reorganized base on the above principles. Functions like `xe()`, `ye()` and `compute()` have a minimum number of inputs based on (1) and all return a single value. For the first two functions the values are scalars while for the last one is an associative array which is processed by the printing `printA1()` function. The parametrized routines can be used to build a library to process any 2R manipulator in PHP.

```

<?php
function xe($x0,$l1,$l2,$fi1,$fi2) {
    return
    $x0+$l1*cos(deg2rad($fi1))+$l2*cos(deg
2rad($fi2));
}

function ye($y0,$l1,$l2,$fi1,$fi2) {
    return
    $y0+$l1*sin(deg2rad($fi1))+$l2*sin(deg
2rad($fi2));
}

function compute($filst, $filend,
$filstep,$fi2st, $fi2end, $fi2step) {
    $l1=1;$l2=1;$x0=0;$y0=0;$l=1;

    for($fi1=$filst;$fi1<=$filend;
$fi1+=$filstep) {
        for($fi2=$fi2st;$fi2<=$fi2end;
$fi2+=$fi2step) {
            $xyea[] = array("fi1" => $fi1, "fi2"
=> $fi2, "xe" =>
xe($x0,$l1,$l2,$fi1,$fi2), "ye" =>
ye($y0,$l1,$l2,$fi1,$fi2));
        }
    }
    return $xyea;
}

function printA1($xyea) {
    $i=0;
    printf("        fi1            fi2            xe
ye\n");
    foreach ($xyea as $xye) {
        printf("%3d    %6.2f    %6.2f    %10.5f
%10.5f\n", $i++, $xye['fi1'], $xye['fi2']
, $xye['xe'], $xye['ye']);
    }
}

```

```
function main() {
    printAl(compute(0, 360, 60, 0, 360,
60));
}
main();
?>
```

3. CONCLUSIONS

The above approach can suffer deeper abstraction and the computation parts can be generalized to a nR (3R, 4R, ...) manipulator with the help functions using simple input parameters and computations with data transfers based on indexed and associative arrays. Still, the flexibility provided to achieve a more advanced reuse is not meet as in the case of routines the only adaptability stands in passing more and different arguments which traps us to the classical “Reuse or Redo” [6] dilemma.

4. REFERENCES

- [1] Rasmus Lerdorf, Kevin Tatroewith, Bob Kaehms and Ric McGredy, *Programming PHP*, O’Reilly, 2002, p. 507, ISBN 1-56592-610-2.
- [2] ANTAL, Tiberiu Alexandru. *A review of the PHP server-side scripting language compared to C, C++ and Java for numerical engineering applications*. ACTA TECHNICA NAPOCENSIS - Series: APPLIED MATHEMATICS, MECHANICS, and ENGINEERING, v. 66, n. 1, 2023. ISSN 2393–2988.
- [3] https://www.dartmouth.edu/basicfifty/basicmanual_1964.pdf
- [4] ANTAL, Tiberiu Alexandru, *Visual BASIC pentru ingineri*, Editura RISOPRINT, 2003, p.244, ISBN 973-656-514-9.
- [5] <https://atom.io/>
- [6] Bertrand Meyer, *Object-oriented software construction*. Prentice Hall PTR, 1997, p.1254, ISBN 0-13629155-4.
- [7] ANTAL, T.A., *Limbajul C ANSI*, Editura RISOPRINT, 2001, p. 253, ISBN 973-656-065-1.
- [8] ANTAL Tiberiu Alexandru, *Proiectarea paginilor Web cu HTML, VBScript și ASP - ediția a II-a*, Editura RISOPRINT, 2006, p.264, ISBN 973-751-349-5.
- [9] ANTAL Tiberiu Alexandru, *Addendum modification of spur gears with equalized efficiency at the points where the meshing stars and ends*, MECHANIKA, Issue: 6 , Pages: 480-485, DOI: 10.5755/j01.mech.21.6.12480, 2015.
- [10] Tiberiu Alexandru Antal, Adalbert Antal, Mariana Arghir, *Determination of the addendum modification at helical gears, at the points where the meshing starts and ends, based on the relative velocity equalization criterion*, Proceedings in Applied Mathematics and Mechanics - PAMM Journal, Volume 8, Issue 1, p.10965-10966, 2008;
- [11] VLASE S; Marin, M; Bratu, P; Manea, R; Shrrat, OAO, *Analysis of Vibration Suppression in Multi-Degrees of Freedom Systems*, *Romanian Journal of Acoustics and Vibration*, 2022, Vol. 19, Issue 2, pp.149-156;
- [12] YA; Utomo, ABS; Mitrayana, M; Lelono, D, *Impedance Boundary Conditions on The Optimal Design of the H-Type Cylinder Resonator Using Transmission Matrix Method and Genetic Algorithm*, *Romanian Journal of Acoustics and Vibration*, 2022, Vol. 19, Issue 1, pp.3-12;
- [13] BRATU P, *The performances in acoustics and vibration engineering*, *Romanian Journal of Acoustics and Vibration*, 2021, Vol. 18, Issue 1, pp.2-2.

Generalizare prin parametrizare cu tablouri asociative, în PHP, într-un calcul manipulator

Lucrarea abordează, folosind limbajul PHP, un caz de abstractizare pentru un manipulator 2R folosind rutine și parametri. Dacă sunt îndeplinite anumite condiții de proiectare și abstractizare, pot fi construite rutine generale pentru a permite reutilizarea codului și în alte contexte. Totuși, aceste biblioteci vor acoperi doar un domeniu limitat de manipulatori, deoarece găsirea echilibrului între abstractizare și simplitate este influențat de nivelul de generalitate, contextul de aplicare și de către dependențele specifice ale manipulatorilor investigați.

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