THE DESIGN OF MICROWAVE FILTERS IN MATLAB

Petr Šmíd, Zbyněk Raida

Dept. of Radio Electronics, FEEC, Brno University of Technology

Abstract

In the first part of this contribution, the design of a planar microstrip filter is described. Descriptions are focused in solving the corrections of the size of the microstrip inductors and capacitors. A size correction is implemented in the *Optimization Toolbox* of MATLAB.

In the second part, artificial neural networks (ANN) are used. We briefly describe why neural networks are used, but we mainly describe an approach to building selected ANN. M-files for working with ANN are implemented in the *Neural Network Toolbox* of MATLAB.

Introduction

A program for designing the microwave filters consists of two parts, because the planar filter is designed in two main steps. The first one is devoted to selecting an appropriate low-pas prototype and building a LC filter. The second main step is devoted to finding a suitable microstrip filter (a stepped-impedance low-pas microstrip filter depicted in Fig. 1, e.g.), which is equivalent to the classical LC filter found in the first step. Both steps are in detail described in [1].

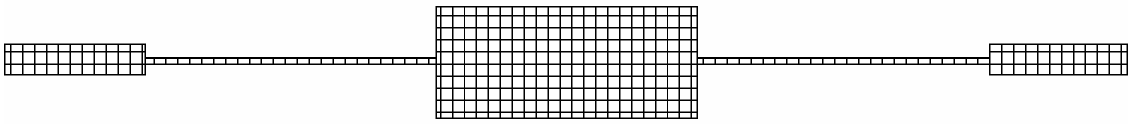
Fig. 1 Layout of a microstrip low-pas filter

During designing the filter by this approach, two problems have to be solved:

Size correction of the microstrip filter components

The first problem is to correct the length of the low-impedance transmission line (a semi-lumped capacitor) and the length of the high-impedance transmission line (a semi-lumped inductor). This problem is described on several examples in [1]. The aim of the correction is to consider the series reactance of the low-impedance line and the shunt susceptance of the high-impedance line. Therefore the model of the low-pas filter (Fig. 1) seems as depicted in Fig. 2.

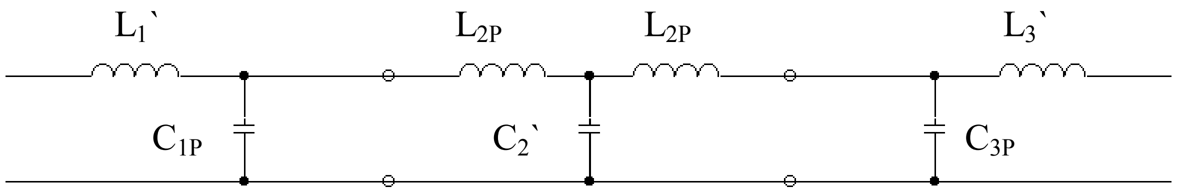

Fig. 2 Lumped model of the filter depicted in Fig. 1

In Fig. 2, the symbols with index P mean the undesirable inductance (or capacitance). The symbols with comma mean the desired inductance or capacitance. In order to consider the undesirable parameters of reactive and susceptive components, we have to come true the next condition: the sum of the inductance of the inductor L_1’ and the inductance of the inductor L_{2P}

has to be equal to the inductance L_1 which is calculated when the classical low-pas filter is designed. Both the rule for the capacity of the low-impedance line and the rule for the last high-impedance line are similar.

Using [1], we are able to generalize the problem for the higher order filters. The equations for the length correction of the third order low-pas filter (where the inductance of the first microstrip inductor is different from the inductance of the second microstrip inductor, Fig. 1) are derived. The derivation approach is given in [2] using [1]:

$$\omega_c L_1 = Z_{0L} \sin\frac{2\pi l_{L1}}{\lambda_{gL}} + Z_{0C} \tan\frac{\pi l_{C2}}{\lambda_{gC}} \tag{1}$$

$$\omega_c L_3 = Z_{0L} \sin\frac{2\pi l_{L3}}{\lambda_{gL}} + Z_{0C} \tan\frac{\pi l_{C2}}{\lambda_{gC}}. \tag{2}$$

$$\omega_c C_2 = \frac{1}{Z_{0C}} \sin\frac{2\pi l_{C2}}{\lambda_{gC}} + \frac{1}{Z_{0L}} \tan\frac{\pi l_{L1}}{\lambda_{gL}} + \frac{1}{Z_{0L}} \tan\frac{\pi l_{L3}}{\lambda_{gL}} \tag{3}$$

In the equations (1) to (3), ω_c means the cutoff frequency, L_1 and L_3 are the desired inductivity of the microstrip inductors, C_2 is the desired capacity of the microstrip capacitor, l_{L1} and l_{L3} mean the length of the microstrip inductors, l_{C2} is the length of the microstrip capacitor, λ_{gL} is the wavelength in the high-impedance components, λ_{gC} is the wavelength in the low-impedance components, Z_{0L} means the wave impedance of the microstrip inductor and Z_{0C} means the wave impedance of the microstrip capacitor.

Exploitation of the artificial neural network to the order estimation of the filter

The second problem is to estimate the order of the low-pas prototype filter with Bessel response. Since we don't know the formula for determining the order of Bessel filter [3], we try to use several types of artificial neural networks to estimate the order of the filter with another response (filter with Chebyshev response). If some of ANNs seem to be proper to estimate the order of the Chebyshev filter, we use this ANN to estimate the order of the filter with Bessel response.

Artificial neural networks are electronic systems (software or hardware ones), which structure is similar to a human brain [4]. ANN consists of a set of neurons (Fig. 3), which are organized to layers [4], [5]. Each neuron multiplies an input number *p* by a weight *w*, adds a constant *b* to the product, and finally, the result is *processed* by a non-linear function.

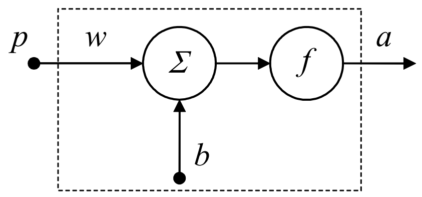

Fig. 3 Neuron

Layers of neurons are mutually connected in order to join output of all the neurons in a preceding layer and the inputs of all the neurons in the succeeding layer via weights. *w*. The smple feed-forward ANN containing four layers with two neurons in each layer is depicted in Fig. 4.

Weights between neurons are set during the training process [4], [5]. The trained ANN provides results very quickly, and therefore, it can be exploited to the solution of problems, which CPU-time demands are very high.

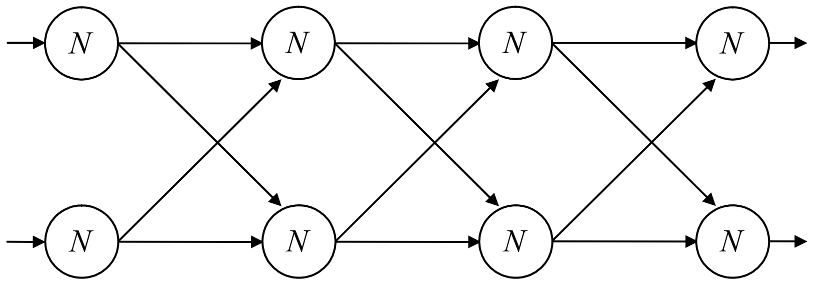

Fig. 4 Mutual connection of neurons in ANN

Implementation in MATLAB

Size correction of the microstrip filter components

In order to correct the size of the higher order low-pas microstrip filter, we have to know several more equations except (1) to (3): equations for the filter connected as Π-network are derived in [2], the other equation for correcting the size of the couple LC is given in [1]. The equations for correcting the size of the components of the symmetrical microstrip filter ($L_1 = L_3$) are two instead of three ones (1) to (3), and these equations are presented in [1]. Knowing all the necessary formulas, we are able to compute the corrected lengths of the components of any even-order microstrip low-pass filter by multiple finding the corrected sizes of the couple LC [1]. If the order of the filter is an odd number bigger than three, we realize the correction of sizes for first three microstrip components of the filter, and further, we continue as in the even-order filter case described. MATLAB implementation of this approach for filter connected as T-network follows:

```
if lower( Zapoj) == 't'              % if the filter is connected in T-network
   if (n/2) == round( n/2)           % even order of LPF – counting the correction of couples LC
      for ii = 1:2:n                 % for all couples LC
         lL_orig = lL(ii);           % original lengths which has to be corrected
         lC_orig = lC(ii+1);
         L_des  = L(ii);             % desired inductivity
         C_des  = C(ii+1);           % desired capacity
         [lL_cor(ii), lC_cor(ii+1)] = …  % calling the function for correction of LC couple
            length_cor2( lL_orig, lC_orig, L_des, C_des, lambda_gL, lambda_gC, Z0L, Z0C, fc);
      end;
      lL_cor(ii+1) = 0;              %  ekeing out the vect. lL_cor to have the same length as lC_cor
   end;
   if (n/2) ~= round( n/2)           % odd order – counting the correction for triplet  L1, C2 and L3
      lL1_orig = lL(1);              % original lengths which has to be corrected
      lC2_orig = lC(2);
      lL3_orig = lL(3);
      L1_des  = L(1);                % desired inductivity L1
      C2_des  = C(2);                % desired capacity C2
      L3_des  = L(3);                % desired inductivity of L3
      [lL_cor(1), lC_cor(2), lL_cor(3)] = ...        % calling the function for correction of triplet L1, C2, L3
         length_cor3T( lL1_orig, lC2_orig, lL3_orig, L1_des, C2_des, L3_des, lambda_gL,
         lambda_gC, Z0L, Z0C, fc);
```

```
        for ii = 4:2:n                      % for the filter branch number 4 to n – correction of couples
            lC_orig = lC(ii);               % original lengths which has to be corrected
            lL_orig = lL(ii+1);
            C_des   = C(ii);                % desired capacity of next couple LC
            L_des   = L(ii+1);              % desired inductivity of next couple LC
            [lL_cor(ii+1), lC_cor(ii)] = ... % calling the function for correction of LC couple
                length_cor2( lL_orig, lC_orig, L_des, C_des, lambda_gL, lambda_gC, Z0L, Z0C, fc);
        end;
    end;
end;
```

In the above source code, the size correction of the microstrip filter components is computed by these functions: *length_cor2* (correction for the couple LC), *lenght_cor3T* (corrects the size of a microstrip LCL-network). The function *lenght_cor3T* founds: the length of the first high-impedance line l_{L1}, the length of the low-impedance line l_{C2} and the length of the second high-impedance line l_{L3} according to (1) to (3). Since the desired variables are at the right side and we know the original values of all three variables, we can use an optimization process [4] solicited in the *Optimization Toolbox* in MATLAB. We build a cost function and we find its minimum. One of possible forms of the cost function can be similar as described in [4]:

$$Q = w_1\left(L_1 - L_{1des}\right)^2 + w_2\left(C_2 - C_{2des}\right)^2 + \left(L_3 - L_{3des}\right)^2 \qquad (4)$$

In (4), symbols L_1, C_2 and L_3 has the same meaning as in (1) to (3), L_{1des}, C_{2des} and L_{3des} are the desired parameters of the classical filter designed in the first main step. The weights are used to correct the order of the addends in (4).

For example, we describe the function *lenght_cor3T*. This function combines optimized parameters (MATLAB variables lL1_or, lC2_or and lL3_or correspond to physical lengths l_{L1}, l_{C2} and l_{L3}) into the vector of original conditions x0. And then, this function calls the function *fminunc*, which finds the minimum of a function of several variables and is in detail described in the help of the *Optimization Toolbox:*

```
x = fminunc( 'objective3T', x0, options, L1, C2, L3, lambda_gL, lambda_gC, Z0L, Z0C, fc);
```

The first parameter 'objective3T' is the cost function stored in its own m-file. The second parameter x0 is the vector containing original values (computed as described in [1]) of the optimized parameters. The next parameter, options, controls the optimization process. The other parameters are transferred to the optimized function objective3T.

The function *objective3T* decomposes the original values from the vector and takes its absolute value to get no negative values during the optimization process. In the following steps, we compute the critical angular frequency, weights for optimized parameters, current values of inductivity and current values of capacity. Finally, we compute the value of the cost function (4). The body of the function *objective3T* follows:

```
x = abs( x);                                  % elimination of negative results
lL1 = x(1); lC2 = x(2); lL3 = x(3);           % expansion of optimized variables
omega = 2*pi*fc;                              % angular critical frequency
w1 = ( L3_des/L1_des)^2;                      % weight for L1
w2 = ( L3_des/C2_des)^2;                      % weight for C2
L1 = ( Z0L * sin( 2*pi*lL1 / lambda_gL)       % actual inductivity L1 according to (1)
    + Z0C * tan( pi*lC2 / lambda_gC)) / omega
```

```
C2 = ( 1/Z0C * sin( 2*pi*lC2 / lambda_gC)            % actual capacity C2 according to (3)
    + 1/Z0L * tan( pi*lL1 / lambda_gL)
    + 1/Z0L * tan( pi*lL3 / lambda_gL)) / omega;
L3 = ( Z0L * sin( 2*pi*lL3 / lambda_gL)              % actual inductivity L3 according to (2)
    + Z0C * tan( pi*lC2 / lambda_gC)) / omega;
Q = 1e+18 * ( w1*(L1 - L1_des)^2                     % the value of the minimized cost function
    + w2*(C2 - C2_des)^2
    + (L3 - L3_des)^2 );
```

The value of the cost function is multiplied by 10^{18} to reach sufficient precision of the optimization process, because the inductance and the capacitance reach too small values. There are some boundaries in options setting for the function *fminunc*, which contain the level for the minimal difference of the cost function Q, but it is easier to multiply the value of the cost function Q by 10^{18} than setting the boundaries and control parameters of the optimization.

Exploitation of the artificial neural network to the order estimation of the filter

The order of the filter depends on four parameters: the type of approximation of the transfer function, the minimal desired attenuation in the stop-band $A_{\min}$, the maximal ripple $A_{\max}$ in the pass-band and the selectivity factor k [3]. Considering the filter with one type of the approximation of the transfer function and keeping one of above numerical parameters constant (e. g. $A_{\max}$), we need ANN with two input neurons and with one output neuron.

We create ANN for the estimation of the Chebyshev filter order with pass-band ripple $A_{\max} = 3$ dB in the next example. First, we have to compose the set of training patterns [4], [5]. The training set consists of a matrix of input patterns and of desired responses of ANN. The matrix of the input patterns includes the combinations of parameters k, $A_{\min}$ a $A_{\max}$. For example, for the values of the selectivity factor k = {1,2; 1,6; 2,0} and for the values of minimal desired attenuation $A_{\min}$ = {10; 15; 20} dB, the training set contains nine patterns (Tab. I).

Tab. I: Set of training patterns

k	[-]	1,2	1,6	2,0	1,2	1,6	2,0	1,2	1,6	2,0
Amin	[dB]	10	10	10	15	15	15	20	20	20
Amax	[dB]	3	3	3	3	3	3	3	3	3

No matter the elements of the vector Amax are of the value, and therefore the vector is not used to train the ANN, we exploit Amax for the function *rad* because this function operates with vectors. Thanks to this fact, we save a cycle in the MATLAB algorithm. The function *rad* computes the order of a filter in accordance with [3] or this function finds the order of the Bessel filter in the table, which is set together using the program NAF for designing electrical filters.

We call the function *rad* with the input in Tab. I. So we get the vector R of nine values. Each value is the order of the filter in one column in Tab. I. Since the output layer of ANN contains the neurons with the output function *logsig* (the output value has to lay in the interval between zero and one – we need a positive number of the order), the vector of the desired output of the ANN has to be normalized. In the next step, we create the variable funkce containing the cells with information about the output function of the neurons in the hidden and the output layer. Inside the ANN, in the hidden layers, we use *tansig* as an output function of each neuron. The function *tansig* gives a number from the interval between minus one and plus one. In the next example, the vector vrstvy contains three digits meaning that ANN has two hidden layers and one output layer. Two neurons are in the first hidden layer, three neurons are in the second hidden layer, and in the output layer, there is one neuron only. The example follows:

```
vrstvy = [ 2 3 1]                          % the number of the neuron in the layers of ANN
P = [k; Amin;];                            % matrix of input learning patterns
R = rad( k, Amax, Amin, Aprox);            % vector of values of the order of filters in Tab. I
Tmax = max( R);                            % maximal order of the filter – normalization constant
T = R / Tmax;                              % desired response of ANN
for ii=1:length( vrstvy)-1;                % for all hidden layers of ANN
   funkce{ii} = 'tansig';                  %   at the output of neurons are tansig functions
end;
funkce{ length( vrstvy)} = 'logsig';       % neurons in the output layer – output function is logsig
```

Further, we create the artificial neural network using the function *newff* implemented in the *Neural Network Toolbox*:

```
net = newff( [min( P(1,:)), max( P(1,:)); min( P(2,:)), max( P(2,:));], vrstvy, funkce, algor);
```

The first input variable of the function *newff* is a matrix, which has two columns and the same number of rows as the number of input neurons is. In the first column, minimal values of input patterns have to be given, and in the second columns, maximal values have to be given. In the last variable algor, the algorithm for learning the ANN has to be saved [4], [5]. Before training the ANN, we set some parameters of training process:

```
net.trainParam.show = 10;                  % showing the state of training after each 10 interations
net.trainParam.epochs = 200;               % stop training when the number of iteration is passed
net.trainParam.goal = 10-4;                % end of training when learning error drops under the limit
```

Finally, we run the learning process and we save the trained network and the normalization constant of the output of the ANN in a file.

```
net = train( net, P, T);                   % learning the ANN
save ('file_name','net', 'Tmax');          % saving learned ANN in a file
```

Now we describe how to visualize the error of the ANN. We have to create a testing set of patterns. This set has to be finer-divided than the training set. The approach to getting the testing set is similar to creating the training set of patterns, described above. Then, we load the stored ANN and the normalization maximum from the file. We place the testing set to the input of the ANN by using the function *sim* implemented in the *Neural Network Toolbox*, and then, we compare the response of the ANN with the values of the filter order, which are determined as described in [3].

```
load ('jmeno_souboru');                    % loading learned ANN and normalizing constant
…                                          % (here has to be created testing set of patterns)
n_vyp = …                                  %   and counting the true order
n_sit = Tmax * sim( net, P);               % response of ANN and de-normalizing
chyba = abs(n_sit - n_vyp);                % difference between the true order and ANN estimation
rel_ch= 100 * chyba ./ n_vyp;              % percentage error
chyba_odh = ceil(n_sit) - ceil(n_vyp);     % the error of order estimation
```

Since we are working with the vectors, but we need to visualize the error depending on two parameters k a $A_{\min}$, we have to reshape the vectors containing the values of the error into a matrices. The matrices may be displayed in a 3D graph. The next example expands only the percentage error of the order estimation by ANN:

```
rel_Amin_k_X_Amax = reshape( rel_ch, length( k_int)*length( Amin_int), length( Amax_int));
for n = 1:length(Amax_int),
    rel_chyba = reshape( rel_Amin_k_X_Amax( :, n), length( k_int), length( Amin_int));
    figure;
    surf(  k_int, Amin_int, rel_chyba');
    title(['Percentage error of feedforward ANN, Passband riple ', num2str( Amax_int(n)), ' dB']);
    xlabel('k [-]');
    ylabel('Amin [dB]');
    zlabel('delta(k, Amin) [%]');
end;
```

The results of order estimation by ANN

The order estimation error of the low-pass prototype with the Chebyshev response, if the feed-forward ANN is used to the estimation, is quite small. The error of the best ANN is lower than 5 % in comparison to the equation, which determines the order [3] (Fig. 5, on the left). This ANN estimated different order in several cases: smaller by one than the formula in [3] in two cases and higher by one in four cases (Fig. 5 - on the right).

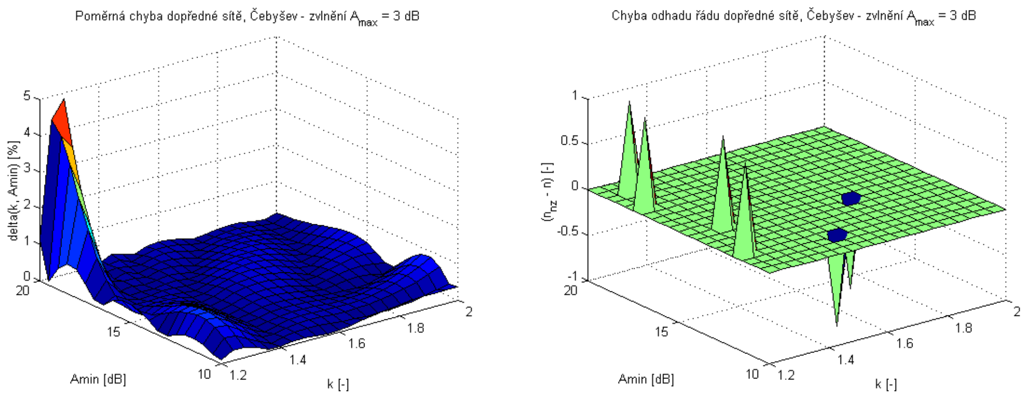

Fig. 5 Percentage error (on the left) and the order estimation error (on the right) of ANN trained for a filter with Chebyshev response

The order estimation error was increased, when the ANN was learned to estimating the order of the filter with Bessel response. In this case, we were not reached satisfactory results: the percentage error rose up to 45 % (Fig. 6, on the left) and the order estimation error rose up to two orders (Fig. 6, on the right).

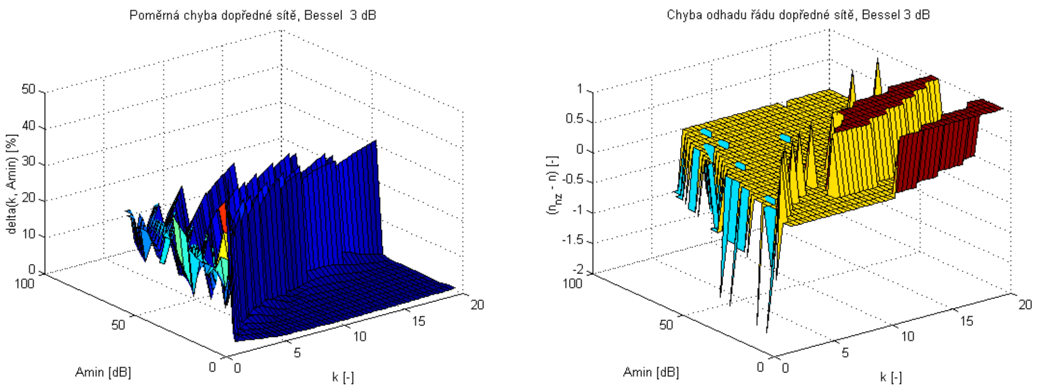

Fig 6 Percentage error (on the left) and the order estimation error (on the right) of ANN trained for a filter with Bessel response

Conclusion

We computed the optimal lengths of the components of a microstrip filter in the first part of this contribution. The example with the given values was not presented, but in our case, the sizes of the microstrip inductors and capacitors are in tens and hundreds of millimeters. Since we know the approximate original sizes of the microstrip components, which are very close to the final sizes of these components, we can use an optimization process from the *Optimization Toolbox* of MATLAB. This approach is a reliable way to get the desired values of mentioned sizes.

The aim of the second part of this contribution was to show how to build an ANN and how to test the error of the ANN for the order estimation of the Bessel low-pass prototype.

The error of the ANN is too high to be suitable for implementing in a program for designing the microstrip filters. The reason of the very high error is given by the fact that the ANN for order estimation of the Bessel filter was learned by using integral values of the order obtained from the program for designing electrical filters NAF instead of learning by a set of real values, as ANN for the order estimation of the Chebyshev filter was.

Acknowledgements

Research described in this contribution was financially supported by the Czech Grant Agency (grants 102/01/0571 and 102/01/0573) and by the Czech Ministry of Education (research programs MSM 262200011 and MSM 262200022).

References

[1] Hong, J. S., Lancaster, M. J., *Microstrip Filters for RF / Microwave Applications*. New York: John Wiley and Sons. 2001. ISBN 0-4713-8877-7

[2] Šmíd P., *Návrh planárních mikrovlnných filtrů. (Design of microwave planar filters)*. Diploma thesis

[3] Dostál T., *Elektrické filtry. (Electric Filters)*. Textbook of Brno University of Technology.2001, ISBN 80-214-0877-4

[4] Černohorský, D., Raida, Z., Škvor, Z., Nováček, Z. *Analýza a optimalizace mikrovlnných struktur. (Analysis and optimization of the microwave structures).* Brno: VUTIUM Publishing, 1999. ISBN 80-214-1512-6.

[5] Demuth, H., Beale, M., *Neural Network Toolbox For Use with MATLAB User's Guide*. (nnet.pdf)

Contact

xsmidp01@stud.feec.vutbr.cz
Department of Radio Electronics
Brno University of Technology
Purkyňova 118
612 00 Brno