Dynamic method in solving linear and non-linear regression equations

1- Abstract

Artificial Intelligence employs various techniques to enhance the performance of machine learning models and train them. Optimization algorithms such as gradient descent play a crucial role in this process. However, university scholars and industrial researchers continue to explore better and more efficient methods to improve gradient descent and accelerate the training of machine learning models.

In this article, we introduce a novel and innovative approach based on dynamical equations to enhance the problem-solving process. We delve into the mathematical and physical aspects of this method, demonstrating how these dynamical equations can significantly improve the performance of gradient descent. Furthermore, we examine the advantages and disadvantages of this approach compared to traditional gradient descent methods.

In conclusion, we report practical experiments and compare the results with conventional gradient descent methods to validate the effectiveness of this new approach. We also provide suggestions for future applications and research directions.

2- Formulation

Assuming we have a model represented by equation (1), we aim to determine the weight matrix of the function, given by equation (2), for various values of features (3), forming a matrix similar to matrix(5).

$$P = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 \quad (1)$$

$$W = \{\alpha_0 \quad \alpha_1 \quad \alpha_2\} \quad (2)$$

$$X = \{1 \quad x_1 \quad x_2\} \quad (3)$$

$$P = \{X\}\{W\}^T \quad (4)$$

We substitute different values for the features into vector (3), forming matrix (5).

$$S = \begin{bmatrix} 1 & 1 & & 1 & 1 \\ a_1 & a_3 & \dots & a_5 & a_7 \\ a_2 & a_4 & & a_6 & a_8 \end{bmatrix} \quad (5)$$

Therefore, equation (1) will be expressed as equation(6).

$$P = S^T W \quad (6)$$

And consider a body similar to the body of shape (1). The dynamic equation for the oscillation of this body is given by equation(7).

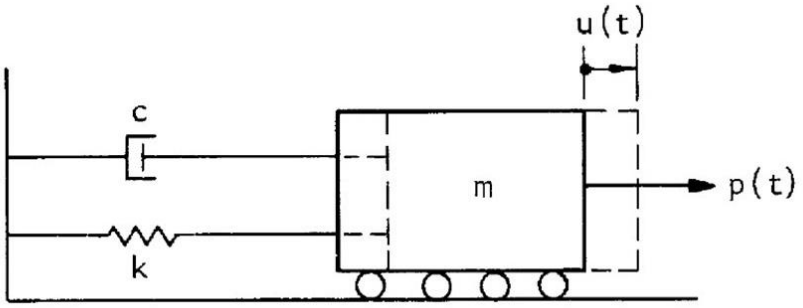


Figure 1: dynamic model

$$P = M\ddot{X} + C\dot{X} + S^T X \tag{7}$$

In the above equation, M represents the mass of the relevant body, C is the damping of the system, S is the stiffness of the spring, and X denotes the displacement values of the body at time t. Considering the artificial mass and artificial damping for the system, equations (6) and (7) together constitute the dynamic state of the system. Therefore, by solving for the values of X in equation (6), we will obtain the weights of our model.

Now, if we rearrange the equation as follows, we can evaluate the error for a gradient descent method in obtaining the weights.

$$R = P - S^T X = M\ddot{X} + C\dot{X} \tag{8}$$

And in the above equation, R represents the error. We consider the changes in the error in two consecutive steps concerning the change in X as follows.

$$\frac{R_{n+1}\text{-}R_n}{\delta = \Delta x} = \lambda \tag{9}$$

And this is in the n+1 step.

$$R_{n+1} = M\ddot{X}_{n+1} + C\dot{X}_{n+1} \tag{10}$$

Now, we express the function X as follows.

$$X(t) = A * \cos(\omega t) \tag{11}$$

So, equation (10) comes in the following form.

$$R_{n+1} = -MA\,\omega^2 \cos(\omega t) - cMA\omega \sin(\omega t) \tag{12}$$

And in the above equation, damping is considered as a coefficient of mass. By substituting this recent relation into equation (9), we obtain equation(13).

$$\frac{-MA\,\omega^2 \cos(\omega t) - cMA\omega \sin(\omega t) - R}{\delta} = \lambda \tag{13}$$

And this is where the value of δ is according to equation(14).

$$\delta = -\tau A\omega \sin(\omega t) \tag{14}$$

And this, by substituting equation (14) into (13) and rearranging it for R, results in equation .(١٥)

$$R = -MA\,\omega^2 \cos(\omega t) - cMA\omega \sin(\omega t) + \lambda\tau A\omega \sin(\omega t) \tag{15}$$

And now, we take the derivative with respect to time on both sides to find the value of λ for the maximum error. After performing the derivative operations, the value of λ will be equal to the expression in equation(16).

$$\lambda = \frac{M(\cos(\omega t)\, c - \omega \sin(\omega t))}{\cos(\omega t)\, \tau} \tag{16}$$

And this, by substituting the recent relation into equation (15) and simplifying it, results in equation(17).

$$A = -\frac{R}{M\omega^2}\cos(\omega t) = -\frac{R}{S^T}\cos(\omega t) \tag{17}$$

And this, considering equation (17) representing the difference in the oscillation range of the spring in two consecutive steps, by substituting it into equation (18), gives us the displacement value for step (n+1).

$$x_{n+1} = x_n - \eta A \cos(\omega t) \tag{18}$$

And this, where η in this equation is the learning rate. The artificial mass of the system is considered to be half the stiffness of the spring, so it is obtained as in equation(19).

$$\mathrm{M} = \frac{\mathrm{S}}{2} \tag{19}$$

And this is where the damping ratio (ω) for the square matrix S is expressed as in equation(20).

$$\omega = \sqrt{\frac{X^T S X}{X^T M X}} \tag{20}$$

And this, if the matrices are non-square, the damping ratio is equal to the expression in equation(21).

$$\omega = \sqrt{\frac{S}{M}} = \sqrt{2} \tag{21}$$

And finally, the cost function will be the average mean square error at each step.

$$cost = \frac{R^T . R}{n} \tag{22}$$

And this, in this equation, n is the number of inputs or the number of elements in the vector (or matrix) R. Keep in mind that if we want to find the weights for multiple equations with one feature each, the weight matrix will transition from a vector to a matrix. Therefore, n will be equal to the number of rows in the matrix R. The initial value of X, considering that the acceleration and velocity are zero at the initial moment, is according to the following relation.

$$x_0 = (S^T)^{-1} P \tag{23}$$

3- Sample

And to demonstrate the performance of the method, several numerical examples will be investigated in comparison with common methods such as gradient descent and the Adam optimizer, and the results will be presented. The program code is written in the Python language.

Input and target datasets:

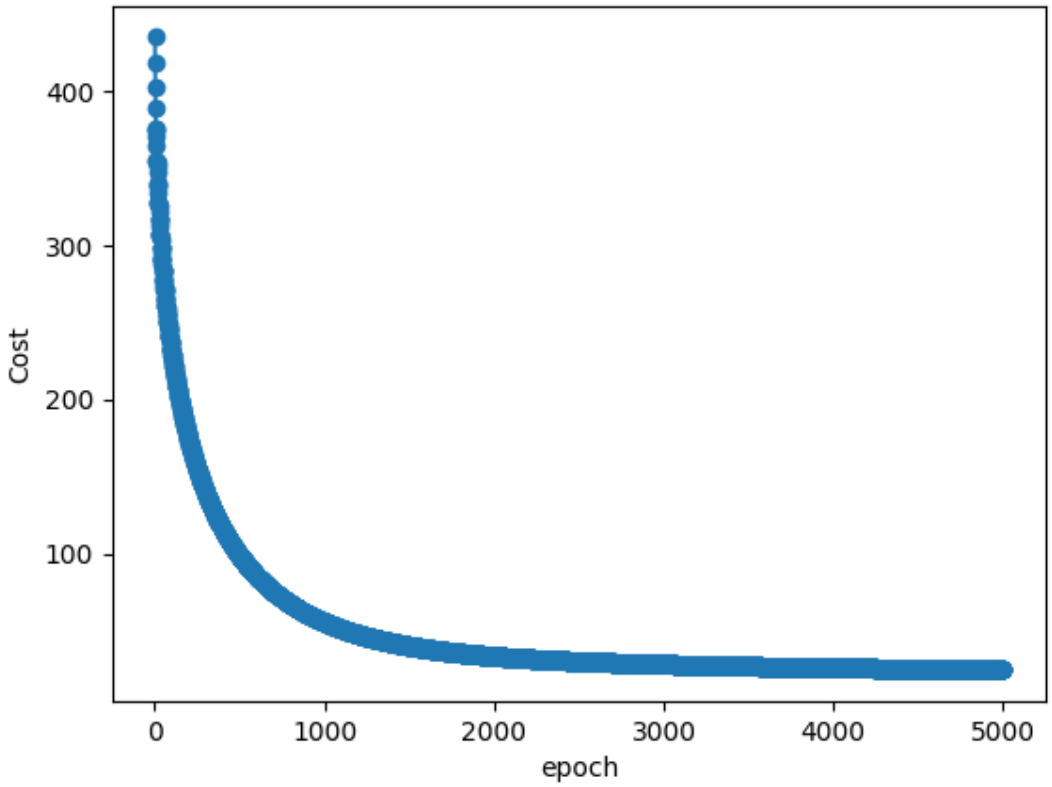
Data point = [(1, 2), (2, 9), (3, 16), (4, 25), (5, 36)]

4- Result

y = a*x+b

And the learning rate in gradient-based methods is set to 0.1. The program stops when the average mean square error over two consecutive epochs becomes less than 0.01 microns. The maximum number of epochs is set to 5000 units.

Final Weight		Time (s)	Cost	Period	Method
a	b				
3.54	5.12	5.35	24.73	5000	Anagrad
-7.59	8.39	2.54	1.52	1274	Adam
-7.59	8.39	2.43	1.52	1594	Gradian
-7.6	8.4	0.01	1.52	1	Dynamic



2-Figure Cost function versus the number of epochs for the Adagrad method.

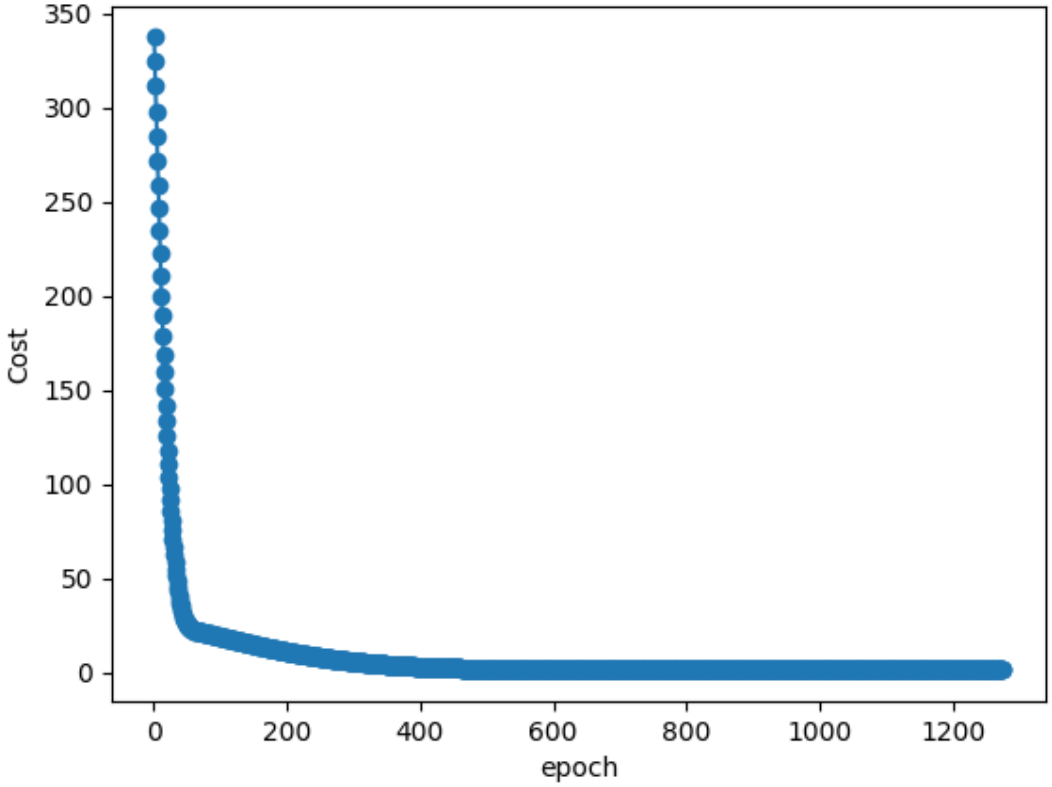


Figure 3: Cost function versus the number of epochs for the Adam method.

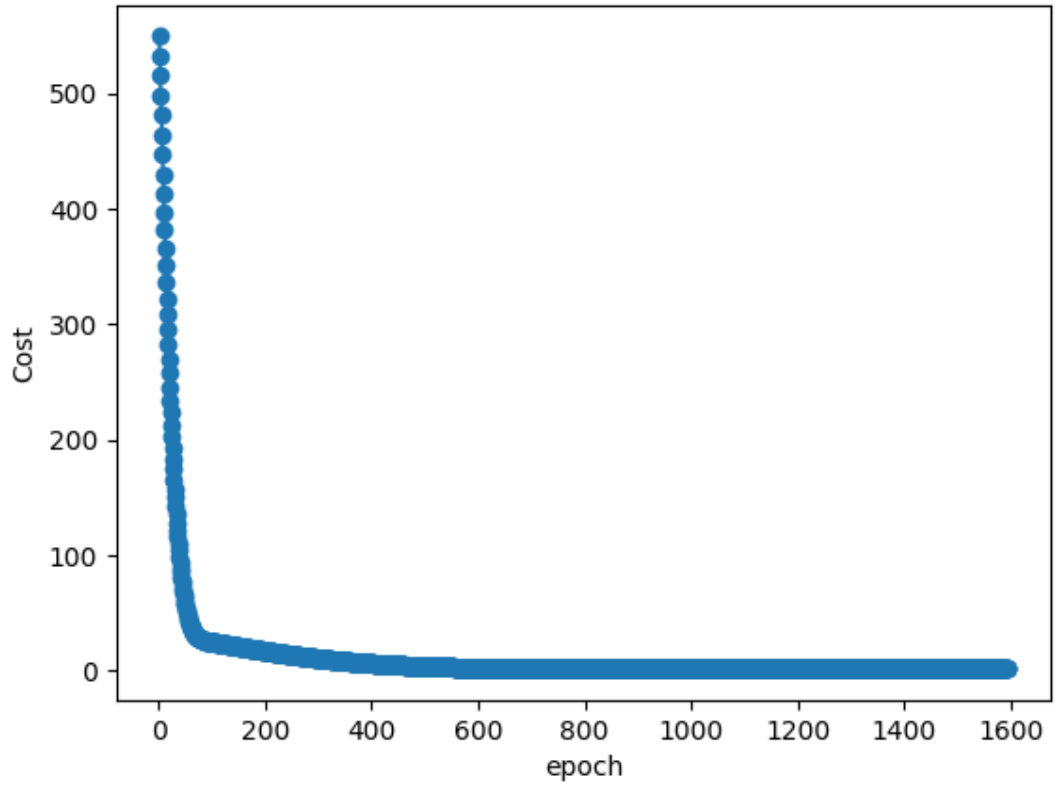


Figure 4: Cost function versus the number of epochs for the Gradient Descent method.

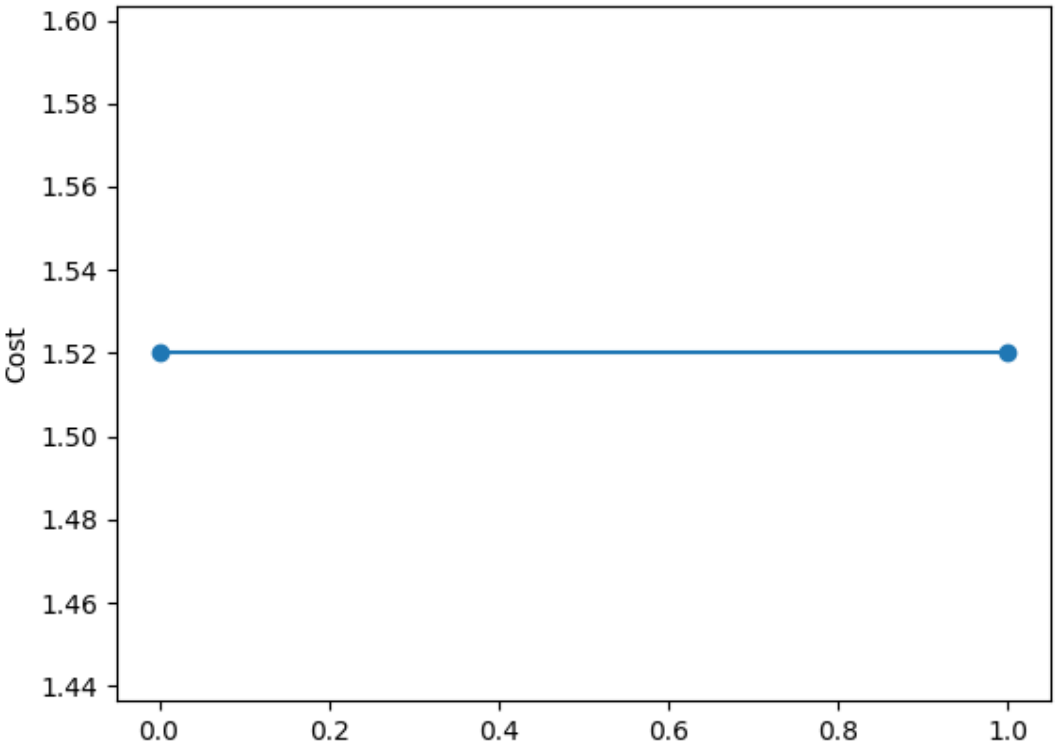


Figure 5: Cost function versus the number of epochs for the Dynamic method.

And this, if we do not consider an initial value for X, the best value for evaluating the learning rate η of this model has been experimentally tested for different learning rate values for better performance.

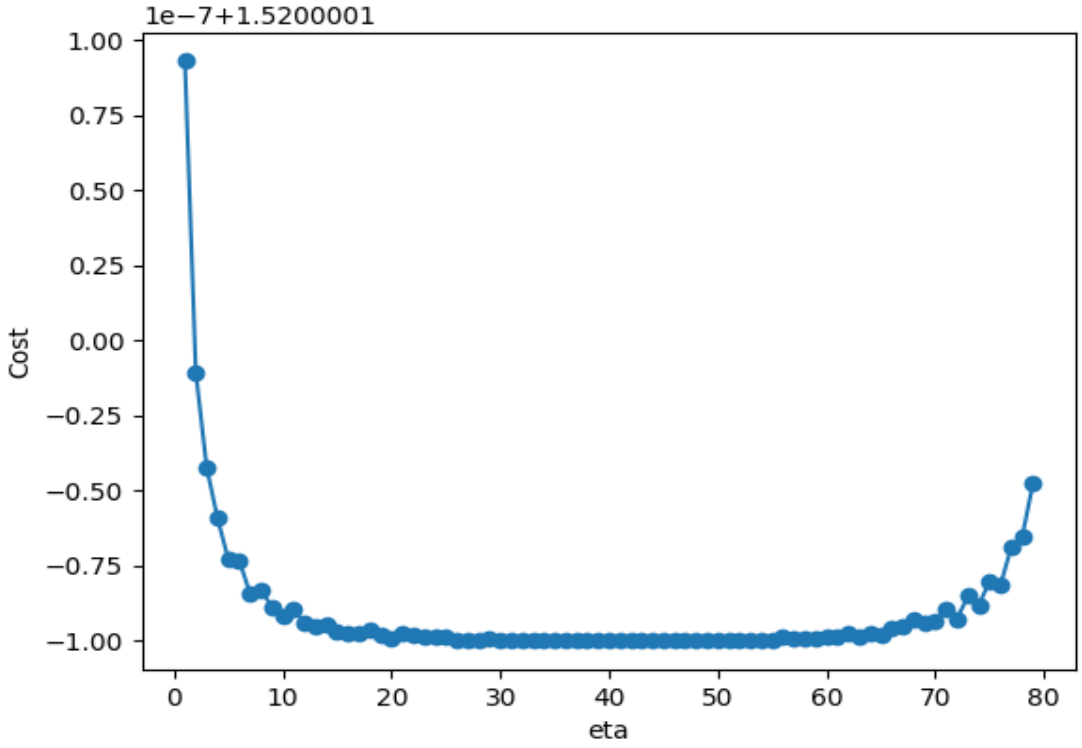


Figure 6: Cost changes versus learning rate (η) variations.

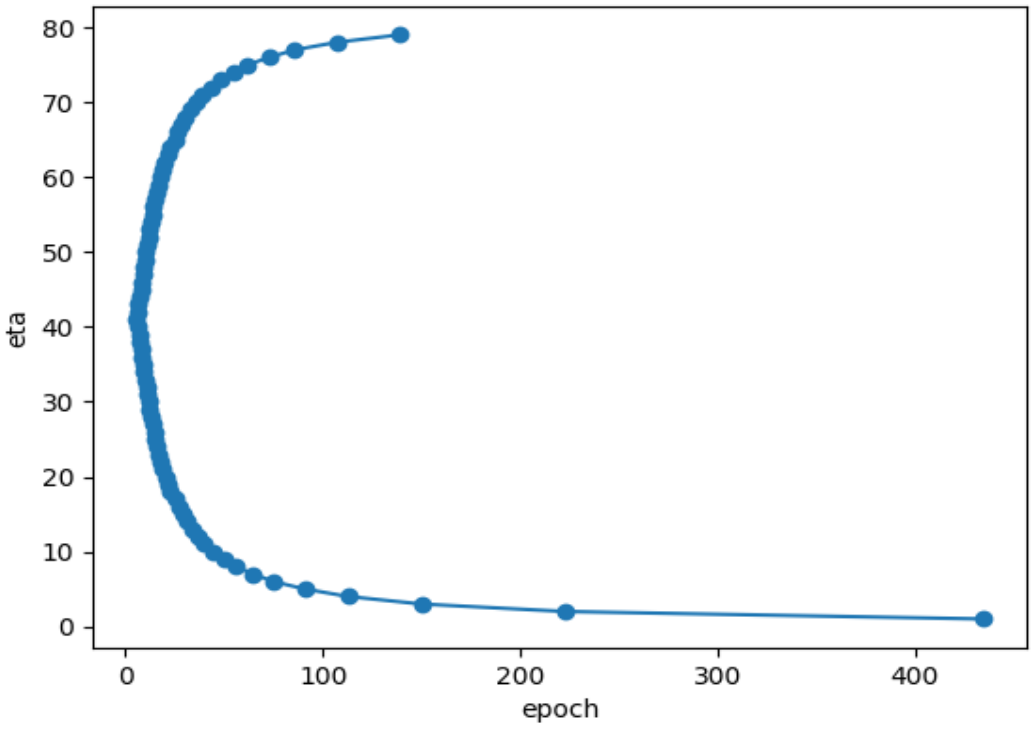


Figure 7: Learning rate changes versus variations in the number of epochs.

5- Conclusion

And this, by examining the performance of the proposed method (dynamic method) against common methods (the best performance among other methods in terms of epochs is for the Adam optimizer), the introduced method demonstrated 1274 times or 100% better performance in terms of epochs. Additionally, in the comparison of execution time, it exhibited 243 times or 99.58% better performance compared to the best method in execution time (gradient descent).

The mentioned method is also applicable for solving non-linear problems, showing good speed in solving such problems. Examining the learning rate, the method performs slower for small learning rates, but still yields better results compared to other methods. For large learning rates, the method converges to divergence for learning rates higher than 80. The best value for the learning rate is 42, achieving the minimum number of epochs and the best cost value. It's worth noting that in the gradient descent method, it diverged for a learning rate of 0.1, and a learning rate of 0.01 was used for it.

A drawback of the introduced method is the use of the matrix inverse, which requires using a pseudo-inverse for non-square matrices. This method is suitable for invertible matrices.

6- Reference

1. Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press Cambridge.
2. Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.
3. Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. Nature, 323(6088), 533-536.
4. Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). Deep learning (Vol. 1). MIT press Cambridge.
5. Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. Science, 349(6245), 255-260.

Python:

https://github.com/systbs/dynamic