Implementing RC network or ESN:

Example for AMOS (forward model)

MakefileConf:

Add this:

```
AMOSIIUTIL  = $(GOROBOTS)/controller/utils


FILES      += main \
                  $(AMOSIIUTIL)/esn-framework/networkmatrix \

INC        += -I$(AMOSIIUTIL)
```

Controller:

1) Add this part in beginning of the file.cpp

```
//Add ENS network--(1)
#include <esn-framework/networkmatrix.h>

//-----ESN network-----//

ESNetwork * ESN;
float * ESinput;
float * ESTrainOutput;
```

2) Add this part in your constructor:

```
NeuralLocomotionControlAdaptiveClimbing::NeuralLocomotionControlAdaptiveClimbing()
{
//-------------------------Add ENS network--
(2)----------------------------------//
  //Setting ENS parameters
  ESN = new ESNetwork(1/*no. input*/,1 /*no. output*/,250 /*rc hidden neurons*/,
false /*feedback*/, false /*feeding input to output*/, 0 /*leak = 0.0*/, false
/*IP*/);

  ESN->outnonlinearity = 0; // 0 = linear, 1 = sigmoid, 2  = tanh: transfer
function of an output neuron
  ESN->nonlinearity = 2; // 0 = linear, 1 = sigmoid, 2  = tanh: transfer function
of all hidden neurons
  ESN->withRL = 2; // 2 = stand ESN learning, 1 = RL with TD learning

  ESN->InputSparsity = 0; // if 0 = input connects to all hidden neurons, if 100 =
```

```
input does not connect to hidden neurons
  ESN->autocorr = pow(10,3); // set as high as possible, default = 1
  ESN->InputWeightRange = 0.15; // scaling of input to hidden neurons, default
0.15 means [-0.15, +0.15]
  ESN->LearnMode = 1;//RLS = 1. LMS =2
  ESN->Loadweight = false; // true = loading learned weights
  ESN->NoiseRange = 0.001; //
  ESN->RCneuronNoise = false; // false = constant fixed bias, true = changing
noise bias every time

  ESN->generate_random_weights(50 /*10% sparsity = 90% connectivity */, 0.95
/*1.2-1.5 = chaotics*/);

  //Create ESN input vector
  ESinput = new float[1];
  //Create ESN target output vector
  ESTrainOutput = new float[1];

  //Initial values of input and target output
  for(unsigned int i = 0; i < 1; i++)
  {
    ESinput[i] = 0.0;
  }

  for(unsigned int i = 0; i< 1; i++)
  {
    ESTrainOutput[i] = 0.0;

  }

  //--------------------------Add ENS network--
(2)----------------------------------//
}
```

3) Add this part in your destructor:

```
NeuralLocomotionControlAdaptiveClimbing::~NeuralLocomotionControlAdaptiveClimbing(
){

  //----- ESN objects garbage collection ---- //

  delete []ESN;
  delete []ESinput;
  delete []ESTrainOutput;
}
```

4) Add this part in your step():

```
std::vector<double> NeuralLocomotionControlAdaptiveClimbing::step_nlc(const
std::vector<double> in0, const std::vector<double> in1){

   //------------Add ESN training (3)---------------------------------//

  bool learn;
  learn = true;
  if(global_count>1000)//100)
```

```
    learn = false;

  ESTrainOutput[0]= reflex_R_fs.at(0); //Training output (target function)
  ESinput[0] = m_pre.at(CR0_m/*6*/);// Input
  ESN->setInput(ESinput, 1/* no. input*/);
  ESN->takeStep(ESTrainOutput, 0.9/*0.9*RLS/ /*0.00055/*0.0005*/
/*0.0055*//*1.5*//*1.8*/, 1 /*no td = 1 else td_error*/, learn/* true= learn,
false = not learning learn_critic*/, 0);

  //temp = ESN->outputs->val(0, 0);
  fmodel_cmr_output_rc.at(0) = ESN->outputs->val(0, 0);
  //output_expected_foot = ESN->outputs->val(0, 1) //second output
  //output_expected_foot = ESN->outputs->val(0, 2) //third output

  //ESN->endweights;
  //------------Add ESN training (3)--------------------------------//


}
```

Example for Nimm4ii (TD learning, RC_critic)

Controller:

1) Add this part in beginning of the file.cpp

```
#include <esn-framework/networkmatrix.h>

//-----ESN network-----//

ESNetwork * ESN, * ESN_actor;

float * ESinput;
float * EAinput;
float * EATrainOutput;
float * ESTrainOutput;
```

2) Add this part in your constructor:

```
ACICOControllerV14::ACICOControllerV14(const ACICOControllerV14Conf& _conf)
: AbstractController("ACICOControllerV14", "$Id: "), conf(_conf)
{
      ESN = new ESNetwork(5/*+1*/,1,100, false, false, 0, false);

      //********************* RC Parameters ********************
      ESN->InputSparsity = 50;
      ESN->InputWeightRange = 0.5;
      ESN->LearnMode = 2;
      ESN->Loadweight = false;
      ESN->NoiseRange = 0.001;
```

```
	ESN->RCneuronNoise = true;
//	ESN->withRL = 1;

	//******************** RC Parameters ********************

	ESN->generate_random_weights(10 /*90*/, 0.95);

	ESN->outnonlinearity = 2;
	ESN->nonlinearity = 2;

	ESinput = new float[5];

	/* initialize inputs to 0 */
	for(unsigned int i = 0; i < 5; i++)
	{
		ESinput[i] = 0.0;

	}

	ESTrainOutput = new float[1]; // single output neuron

	ESTrainOutput[0] = 0.0;
```

3) Add this part in your destructor:

```
	delete []ESN;
	delete []ESinput;
	delete []ESTrainOutput;

	delete []ESN_actor;
	delete []EAinput;
	delete []EATrainOutput;
```

4) Add this part in your step():

```
void ACICOControllerV14::step(const sensor* x_, int number_sensors, motor* y_, int
number_motors){



	if (ESN_critic){


				ESTrainOutput[0]= acum_reward;

				ESinput[0] = xt[_ias0] ; //+ gauss();
				ESinput[1] = xt[_ias1] ; //+ gauss();

				ESinput[2] = rt;

				ESinput[3] = xt[_ias2];
				ESinput[4] = xt[_ias3];

				if (!ESN) cout<< "critical failure: ESN not loaded"
<<std::endl;
```

```
                        ESN->setInput(ESinput, 5 /*3*/);

                    bool learn = true;

                        if(exp_output[0]< 0.0001) learn = false;

ESN->takeStep(ESTrainOutput, 0.00022 /*0.00055/*0.0005*/ /*0.0055*//*1.5*//*1.8*/,
td_error, learn/*learn_critic*/, 0);

                        Vt = ESN->outputs->val(0, 0) * 50;

                std::cout<<"acum_reward "<<acum_reward<<"\n";

                  }
                  // *********** commented ESN ***************
```