LQR Forward pass

Notation

In this part we consider a normal distribution over $\mathbf{x}$ and $\mathbf{u}$:

$$\begin{pmatrix}\mathbf{x}\\ \mathbf{u}\end{pmatrix} \sim \mathcal{N}(\mu, \mathbf{\Sigma})$$

where

$$\mu = \begin{pmatrix}\mu_{\mathbf{x}}\\ \mu_{\mathbf{u}}\end{pmatrix}$$

$$\mathbf{\Sigma} = \begin{pmatrix}\mathbf{\Sigma}_{\mathbf{x},\mathbf{x}} & \mathbf{\Sigma}_{\mathbf{x},\mathbf{u}}\\ \mathbf{\Sigma}_{\mathbf{u},\mathbf{x}} & \mathbf{\Sigma}_{\mathbf{u},\mathbf{u}}\end{pmatrix}$$

Code

The forward recursion function takes as arguments

- `traj_distr` : The policy object containing the linear gaussian policy
- `traj_info` : This object contains the dynamics

```
def forward(self, traj_distr, traj_info):
```

We get the number of timesteps and the dimensions of the states and the actions from the policy object:

```
T = traj_distr.T
dimU = traj_distr.dU
dimX = traj_distr.dX
```

We use slice syntax so that `sigma[index_x, index_u]` means $\mathbf{\Sigma}_{\mathbf{x},\mathbf{u}}$ etc.

```
index_x = slice(dimX)
index_u = slice(dimX, dimX + dimU)
```

We allocate space for μ and $\mathbf{\Sigma}$ and set the initial values for $\mu_{0,\mathbf{x}}$ and $\mathbf{\Sigma}_{0,\mathbf{xx}}$:

```python
sigma = np.zeros((T, dimX + dimU, dimX + dimU))
mu = np.zeros((T, dimX + dimU))

mu[0, index_x] = traj_info.x0mu
sigma[0, index_x, index_x] = traj_info.x0sigma
```

We iterate over t and compute:

$$\mu_{t,\mathbf{u}} = \mathbf{K}_t \mu_{t,\mathbf{x}} + \mathbf{k}_t$$
$$\mathbf{\Sigma}_{t,\mathbf{x},\mathbf{u}} = \mathbf{\Sigma}_{t,\mathbf{x},\mathbf{x}} \mathbf{K}_t^T$$
$$\mathbf{\Sigma}_{t,\mathbf{u},\mathbf{x}} = \mathbf{K}_t \mathbf{\Sigma}_{t,\mathbf{x},\mathbf{x}}$$
$$\mathbf{\Sigma}_{t,\mathbf{uu}} = \mathbf{K}_t \mathbf{\Sigma}_{t,\mathbf{x},\mathbf{x}} \mathbf{K}_t^T + \mathbf{\Sigma}_{pol,t}$$

```python
for t in range(T):
    mu[t, index_u] = traj_distr.K[t, :, :].dot(mu[t, index_x]) + \
                     traj_distr.k[t, :]

    sigma[t, index_x, index_u] = \
        sigma[t, index_x, index_x].dot(traj_distr.K[t, :, :].T)

    sigma[t, index_u, index_x] = \
        traj_distr.K[t, :, :].dot(sigma[t, index_x, index_x])

    sigma[t, index_u, index_u] = \
        traj_distr.K[t, :, :].dot(sigma[t, index_x, index_x]).dot(
            traj_distr.K[t, :, :].T
        ) + traj_distr.pol_covar[t, :, :]
```

for $t < T$ we compute:

$$\mu_{t+1,\mathbf{x}} = \mathbf{F}_t \mu_t + \mathbf{f}_t$$
$$\mathbf{\Sigma}_{t+1,\mathbf{x},\mathbf{x}} = \mathbf{F}_t \mathbf{\Sigma}_t \mathbf{F}_t^T + \mathbf{\Sigma}_{dyn}$$

```python
if t < T - 1:
    mu[t+1, index_x] = Fm[t, :, :].dot(mu[t, :]) + fv[t, :]

    sigma[t+1, index_x, index_x] = \
        Fm[t, :, :].dot(sigma[t, :, :]).dot(Fm[t, :, :].T) + \
        dyn_covar[t, :, :]
```

After that the loop ends and we return `mu` and `sigma`:

```python
return mu, sigma
```