P1, P2:

1、$\dfrac{\partial R^{-1}p}{\partial R}$

右扰动：

$$\lim_{\phi \to 0} \frac{(R\exp(\phi))^{-1}p - R^{-1}p}{\phi}$$

$$= \lim_{\phi \to 0} \frac{\exp(\phi)^{-1}R^{-1}p - R^{-1}p}{\phi}$$

$$= \lim_{\phi \to 0} \frac{(\exp(\phi)^{-1} - I)R^{-1}p}{\phi}$$

$$= \lim_{\phi \to 0} \frac{(\cancel{I} - \phi^\wedge - \cancel{I})R^{-1}p}{\phi}$$

$$= \lim_{\phi \to 0} \frac{-\phi^\wedge R^{-1}p}{\phi}$$

$$= \lim_{\phi \to 0} \frac{(R^{-1}p)^\wedge \phi}{\phi}$$

$$= (R^{-1}p)^\wedge$$

左扰动

$$\frac{\partial (R^{-1}p)}{\partial \phi} = \lim_{\phi \to 0} \frac{((\exp(\phi)R)^{-1}p - R^{-1}p)}{\phi}$$

$$= \lim_{\phi \to 0} \frac{R^{-1}(I - \phi^\wedge)p - R^{-1}p}{\phi}$$

$$= \lim_{\phi \to 0} - \frac{R^{-1} \phi^\wedge p^\wedge}{\phi}$$

$$= R^{-1} p^\wedge$$

2. $\frac{\partial R_1 R_2^{-1}}{\partial R_2}$

右扰动:

$$= \lim_{\phi_2 \to 0} \frac{\log (R_1 (R_2 \exp(\phi_2))^{-1}) - \log R_1 R_2^{-1}}{\phi_2}$$

$$= \lim_{\phi_2 \to 0} \frac{\log (R_1 \exp(\phi_2)^{-1} R_2^{-1}) - \log R_1 R_2^{-1}}{\phi_2}$$

$$= \lim_{\phi_2 \to 0} \frac{\log (R_1 R_2^{-1} \exp(-R_2 \phi_2)) - \log R_1 R_2^{-1}}{\phi_2}$$

BCH $$= \lim_{\phi_2 \to 0} \frac{\log (\exp(\log (R_1 R_2^{-1}) + J_r^{-1}(R_1 R_2^{-1}) R_2 \phi_2)) - \log R_1 R_2^{-1}}{}$$

$$\overline{\phi_2 \qquad\qquad}$$

$$= \lim_{\phi_2 \to 0} \frac{\cancel{\log(R_1R_2^{-1})} - \left(J_r^{-1}(R_1R_2^{-1}) R_2 \phi_2\right) - \cancel{\log R_1R_2^{-1}}}{\phi_2}$$

$$= -J_r^{-1}(R_1 R_2^{-1}) R_2$$

左扰动:

$$\lim_{\phi_2 \to 0} \frac{\log\left(R_1\left(\exp(\phi_2^\wedge) R_2\right)^{-1}\right) - \log(R_1R_2^{-1})}{\phi_2}$$

$$= \lim_{\phi_2 \to 0} \frac{\log R_1 R_2^{-1} \exp(\phi_2^\wedge) - \log(R_1R_2^{-1})}{\phi_2}$$

BCH

$$= \lim_{\phi_2 \to 0} \frac{\log \exp\left(\log(R_1R_2^{-1}) + J_r(R_1R_2^{-1})(-\phi_2)\right)^\wedge - \log R_1R_2^{-1}}{\phi_2}$$

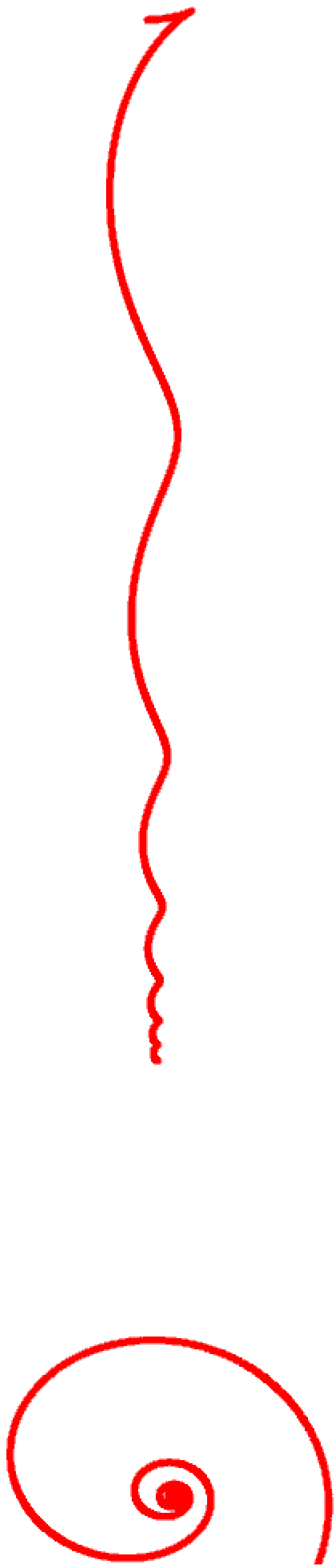
$$= \lim_{\phi_2 \to 0} \frac{-J_r(R_1 R_2^{-1}) \phi_2}{\phi_2}$$

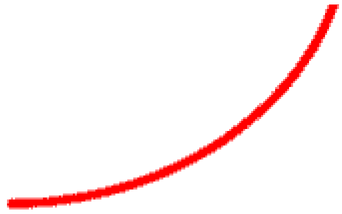
$$= -J_r(R_1R_2^{-1})$$

P3:

```
#include <gflags/gflags.h>
#include <glog/logging.h>

#include "common/eigen_types.h"
#include "common/math_utils.h"
#include "tools/ui/pangolin_window.h"

/// 本节程序演示一个正在作圆周运动的车辆
/// 车辆的角速度与线速度可以在flags中设置

DEFINE_double(angular_velocity, 10.0, "角速度（角度）制");
DEFINE_double(linear_velocity, 5.0, "车辆前进线速度 m/s");
DEFINE_double(gravity, 9.81, "车辆前进线速度 m/s");
DEFINE_bool(use_quaternion, false, "是否使用四元数计算");

int main(int argc, char** argv) {
    google::InitGoogleLogging(argv[0]);
    FLAGS_stderrthreshold = google::INFO;
    FLAGS_colorlogtostderr = true;
    google::ParseCommandLineFlags(&argc, &argv, true);

    /// 可视化
    sad::ui::PangolinWindow ui;
    if (ui.Init() == false) {
        return -1;
    }

    double angular_velocity_rad = FLAGS_angular_velocity * sad::math::kDEG2RAD;
// 弧度制角速度
    SE3 pose;
// TWB表示的位姿
    Vec3d omega(0, 0, angular_velocity_rad);
// 角速度矢量
    Vec3d v_body(FLAGS_linear_velocity, 0, 0);
// 本体系速度
    Vec3d a_world(0, 0, -FLAGS_gravity);
    Vec3d a_body(0, 0, 0);
    const double dt = 0.05;
// 每次更新的时间

    while (ui.ShouldQuit() == false) {
        // 更新自身位置
        v_body += pose.so3().inverse() * a_world * dt;
        Vec3d v_world = pose.so3() * v_body;

        pose.translation() += v_world * dt + a_body * dt * dt / 2.;

        // 更新自身旋转
        if (FLAGS_use_quaternion) {
            Quatd q = pose.unit_quaternion() * Quatd(1, 0.5 * omega[0] * dt, 0.5
* omega[1] * dt, 0.5 * omega[2] * dt);
```

```
            q.normalize();
            pose.so3() = SO3(q);
        } else {
            pose.so3() = pose.so3() * SO3::exp(omega * dt);
        }

        // LOG(INFO) << "pose: " << pose.translation().transpose();
        LOG(INFO) << "v_world: " << v_world.transpose();
        ui.UpdateNavState(sad::NavStated(0, pose, v_world));

        usleep(dt * 1e6);
    }

    ui.Quit();
    return 0;
}
```





P4:
高斯牛顿法：

$(J^T*J) * \delta x = -J^T * r$

在迭代时用一阶的雅各比J^T*J来近似二阶Hessian矩阵。这样做的好处在于雅各比的计算量量级比牛顿法的Hessian矩阵小一个量级。这样的做法缺点在于在距离最优点比较远的状态量时，雅各比的近似质量比较差

Levenberg-Marquardt:

$(J^T*J + \lambda *I) * dx = -J^T * r$

属于Trust Region Method中的一种。LM法就是为了解决GN的缺点:当lambda比(J^T*J)大很多时，迭代的方向就约等于 -J^T * r， 相当于是梯度下降。当lambda比(J^T*J)小时，LM法迭代的方向和GN相当。所以一般来说会在优化开始时选择一个比较大的$\lambda ，根据优化的进度一点点减少\lambda 。